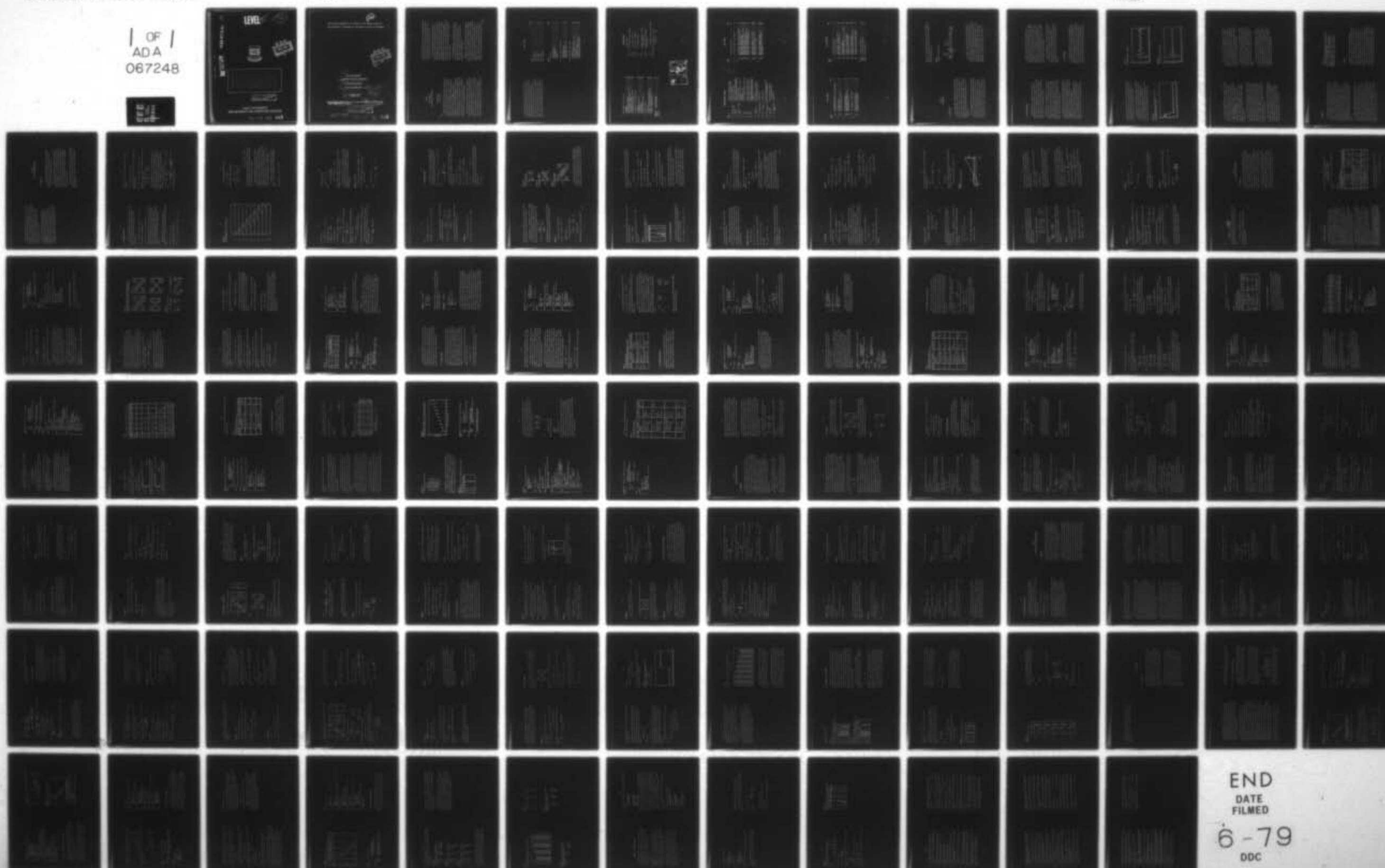
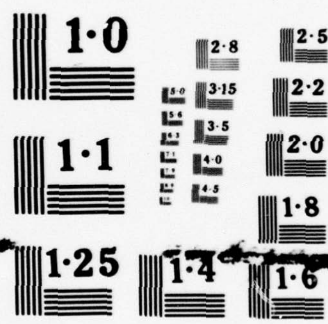
AD-A067 248 YALE UNIV NEW HAVEN CONN DEPT OF COMPUTER SCIENCE F/G 12/1
SPLINE REGRESSION: ALGORITHMS AND LOCAL DEPENDENCE.(U)
DEC 78 J W LEWIS N00014-76-C-0277
UNCLASSIFIED RR-148 NL

1 OF 1
AD A
067248



END
DATE
FILMED
6-79
DDC


1·0
2·8
2·5
5·0
3·15
2·2
3·5
1·1
2·0
4·0
4·5
1·8
1·25
1·4
1·6

YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

(12)

This work was presented to the faculty of the Graduate School of Yale University in candidacy for the Degree of Doctor of Philosophy.



(6)

# SPLINE REGRESSION: ALGORITHMS AND LOCAL DEPENDENCE.

(10) John Winslow/Lewis

(9) Research Report, #148 (14) RR-148

(11) December 1978 (12) 94p.

This work was supported in part by ONR Grant (15) N00014-76-C-0277, and NSF ~~Grant~~-MCS 76-11460

## ABSTRACT

# SPLINE REGRESSION:
# ALGORITHMS AND LOCAL DEPENDENCE

John Winslow Lewis
Yale University

Curve fitting has been an important problem in data analysis and curve design for many years. In curve fitting, for some set of data, the goal is a smooth curve which is close to the data. A variety of drafting techniques (e.g., French curves, draftsman's splines, and flexible curves) and mathematical methods (e.g., polynomial interpolation, least-squares polynomials, and smoothing formulae) have been developed to solve curve fitting problems. Spline regression is a relatively new mathematical curve fitting method which has proved to be useful for moderately accurate (2 to 5 decimal digit) approximations to data which are difficult to approximate by analytic means.

The qualitative behavior of least-squares spline approximations differs significantly from that of most classical approximation schemes in that least-squares splines are highly local. While the value of a polynomial (or any other analytic function) at a point can be determined from its value and derivatives at any arbitrarily distant point, the value of the least-squares spline at any point is almost completely determined by neighboring data. Moreover, the effect of distant data on the least-squares spline's value at a point decreases exponentially with the number of knots separating that data from the evaluation point, and the local approximation error is almost completely determined by the local knot density and local characteristics of the data.

The local dependence properties of spline approximations are derived from the corresponding local dependence properties of the associated linear systems. In this dissertation, a unified theory of local dependence is developed for symmetric, positive definite, block tridiagonal matrices and the associated linear systems. The results include bounds on elements of the inverse, simple local dependence bounds, error bounds for local solutions to linear systems, error bounds for local inverses of matrices, and error bounds for local Cholesky factorizations of matrices.

The local dependence theory for least-squares splines follows directly from the local dependence theory for matrices. The results include simple local dependence bounds, error bounds for local least-squares spline approximations, and local error bounds for least-squares spline approximations.

Algorithms to compute least-squares splines can be significantly more efficient than many classical analytic approximation algorithms. In this dissertation, a detailed analysis of algorithms for computing and evaluating least-squares spline approximations to data is presented. The algorithms are given explicitly in an ALGOL-like language and operation counts are presented. Of particular interest are a fast incremental algorithm for evaluating splines and a limited-storage algorithm for computing piecewise polynomial representations of splines.

This algorithm analysis and the local solution schemes for computing least-squares splines are combined to create an efficient, limited-storage algorithm for least-squares spline data fitting. The algorithm is designed for data fitting and signal processing applications where large quantities of data are processed on-line in small computers. The algorithm scans the data only once, producing the B-spline coefficients for the least-squares spline as the data are scanned. For a fixed relative accuracy, the algorithm requires a fixed amount of storage and a fixed number of operations per data point.

TABLE OF CONTENTS

## STANDARD NOTATION

$K$ -- a generic constant

$D^{\ell}f$, $f^{(\ell)}$ -- the $\ell^{th}$ derivative of the function $f$

$\|\underline{x}\|_{\ell_p}$ -- the $\ell_p$ norm of the vector $\underline{x}$

$\|f\|_{L_p}$ -- the $L_p$ norm of the function $f$

$p_n(t)$ -- a polynomial of degree $n$

$C^r[a,b]$ -- the set of functions with continuous $r^{th}$ derivative on $[a,b]$

$L_p[a,b]$ -- the set of functions with bounded $L_p$ norm on $(a,b)$

$\omega(f,h)$ -- the modulus of continuity of the function $f$

$g(x) = O(f(x))$ -- $g(x)/f(x) \le K$ as $x \to 0$ OR $g(x)/f(x) \le K$ as $x \to \infty$

## NOTATION PECULIAR TO SPLINES

| Page | | |
|---|---|---|
| 13 | $k$ | -- the order (or degree+1) of a spline |
| 13 | $m$ | -- the number of interior break points |
| 13 | $\underline{u} \equiv (u_0,\ldots,u_{m+1})$ | -- the vector of break-points |
| 13 | $\underline{z} \equiv (z_1,\ldots z_m)$ | -- the knot multiplicity vector |
| 13 | $S(k,\underline{u},\underline{z})$ | -- the set of splines of order k with break-point vector $\underline{u}$ and multiplicity vector $\underline{z}$ |
| 13 | $\underline{t} \equiv (t_1,t_2,\ldots,t_{n+k})$ | -- a B-spline knot vector |
| 13 | $S(k,\underline{t})$ | -- the collection of splines of order k with knot vector $\underline{t}$ |
| 13 | $n$ | -- the dimension of $S(k,\underline{t})$ |
| 14 | $N_{i,k}(t)$ | -- the $i^{th}$ B-spline basis function of order k |
| 13 | $s(t)$ | -- a spline function in $S(k,\underline{t})$ or $S(k,\underline{u},\underline{z})$ |
| 14 | $\underline{a}$ | -- a B-spline basis coefficient vector |
| 18 | $\underline{a}^{(\ell)}$ | -- the basis coefficient vector of the $\ell^{th}$ derivative of s(t) expanded as B-splines of order k-$\ell$ |
| 20 | $P_{S(k,\underline{t})}f(t)$ | -- the $L_2$ projection of the function f(t) onto the spline space $S(k,\underline{t})$ |
| 20 | $G \equiv [\,(N_{i,k},N_{j,k})_{L_2}\,]$ | -- the Gram matrix |
| 20 | $\underline{b} \equiv [\,(f,\, N_{i,k})_{L_2}\,]$ | -- the data vector $\underline{b}$ |
| 23 | $N$ | -- the number of points in a set of data |
| 23 | $Y \equiv \{\,(w_\ell,x_\ell,y_\ell) \mid 1 \le \ell \le N\,\}$ | -- A set of weighted data |
| 23 | $P_{S(k,\underline{t})}Y(t)$ | -- the $\ell_2$ projection of the data Y onto the spline space $S(k,\underline{t})$ |
| 127 | $\lvert\underline{t}\rvert \equiv \max_{k\le i\le n} t_{i+1}-t_i$ | -- the mesh width |



## LIST OF TABLES

## LIST OF FIGURES





## LIST OF ALGORITHMS

## Chapter I

## Introduction

### 1.1 Curve Fitting

Curve fitting has been an important problem in data analysis and curve design for many years. For some set of data, the goal is a smooth curve which is close to the data. A variety of drafting techniques [F4] (e.g., French curves, draftsman's splines, and flexible curves) and mathematical methods [B1] (e.g., polynomial interpolation, least-squares polynomials, and smoothing formulae) have been developed to solve curve fitting problems.

While the draftsman's spline has been employed in ship hull design for many centuries [F4] and the method of least-squares has been popular in data analysis since the early nineteenth century [H2], these two techniques were not combined as spline regression until ten to fifteen years ago [B10,P3]. Since then, spline regression has proved to be useful for moderately accurate (2 to 5 decimal digit) approximations to "real-world" data (e.g., Figure 1.1, [R1,p.164]) which are difficult to



approximate by conventional analytic means. However, spline regression has not been a competitive technique for high accuracy approximations to analytic functions [R1,p.163].

FIGURE 1.1

A Parametric Spline Approximation

Spline regression has become a popular data-fitting technique for two major reasons: 1) unlike classical analytic approximations, the resulting approximation is highly local [P1;R1,p.123] and 2) algorithms to compute least-squares splines are significantly more efficient than many classical analytic approximation algorithms (see Table 4.1). This dissertation is a study of these two aspects of least-squares spline approximation: local dependence properties of least-squares spline approximations and algorithms for computing least-squares splines.

## 1.2 Splines and Regression

The draftsman's spline is a traditional (but now obsolete) curve-drawing technique [F4,p.10,30,445]. A thin strip of wood or plastic is secured at a number of points (called knots) by lead weights (called ducks) which a draftsman can manipulate to form the desired curve. The spline follows the curve which minimizes the bending energy of the strip subject to the constraints at the knots.

In general, the shape of the draftsman's curve is difficult to describe explicitly [M1]; but in the limiting case of an infinitely thin strip, the spline can be viewed as a simply supported thin beam [A1,Chapter I;T1,Chapter V] and the resulting curve is a cubic spline function (or piecewise polynomial). Between each pair of knots, the spline function is a (possibly different) cubic polynomial, and at each knot it has at least two continuous derivatives.

The mathematical study of spline functions was begun shortly after World War II by Courant in a study of the finite element method for solving differential equations [C1] and Schoenberg in a paper concerning data approximation [S1]. Subsequently, an extensive theory for spline approximation of functions has been developed (e.g., [A1,G3,S6,S11,P3,B8]). That portion of spline approximation theory which is relevant to this study of spline regression is presented in Chapter II. Results extending the theory to the least-squares approximation of data are included in the final section of Chapter II.



Regression (the method of least-squares) is also an old data-fitting technique [H2]. In the early eighteenth century, Gauss, Legendre, Laplace and others (see [H2] for a review) solved a number of data approximation problems using the method of least-squares. The least-squares (or best $\ell_2$) approximation to any set of data over some function space (e.g., the polynomials of some fixed degree) is the function minimizing the sum-squared approximation error. For example, the least-squares polynomial approximation of degree zero to a set of data is the average (or mean) of that data.

For data with random Gaussian error, a least-squares approximation can be viewed as the most-likely representation for the data in that class of functions [H2]. Consequently, because least-squares approximations are also easy to compute, the method of least-squares has been employed widely in physical and social science applications since 1850 [H2].

## 1.3 Local Dependence

The qualitative behavior of least-squares spline approximations differs significantly from that of most classical approximation schemes because least-squares splines are highly local. While the value of a polynomial (or any other analytic function) at a point can be determined from its value and derivatives at any arbitrarily distant point, the value of the least-squares spline at any point is almost completely determined by neighboring data. Moreover, the effect of distant data on

the least-squares spline's value at a point decreases exponentially with the number of knots separating that data from the evaluation point, and the local approximation error is almost completely determined by the local knot density and local characteristics of the data.

The difference between the local dependence behavior of spline and polynomial regression is illustrated by the approximation of the step function data in Figure 3.1. While the least-squares polynomial oscillates considerably throughout the entire interval (see Figure 3.1c), the least-squares spline is very close in value to the data at any point far from the discontinuity (see Figure 3.1b).

FIGURE 3.1a

Local Dependence: Cubic spline regression with 20 knots and polynomial regression of order 20

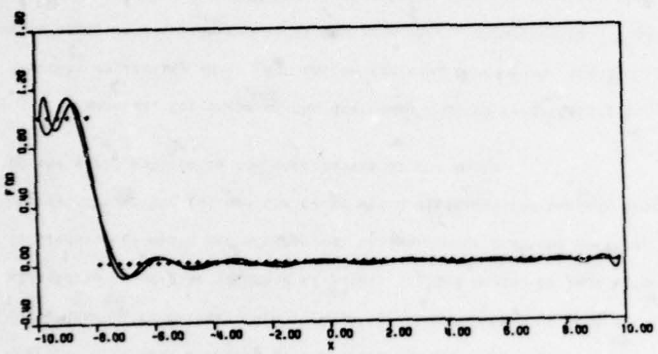





FIGURE 3.2b

Polynomial Regression: (magnified)

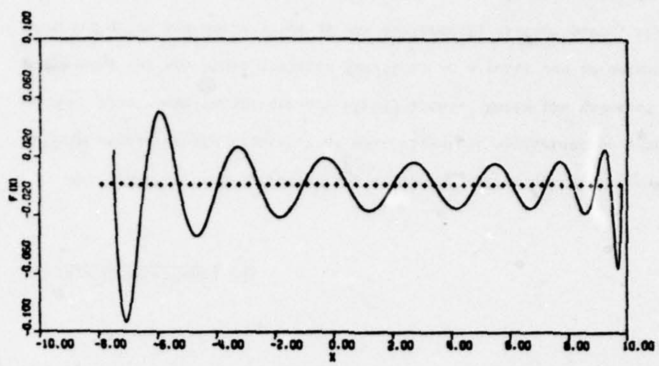
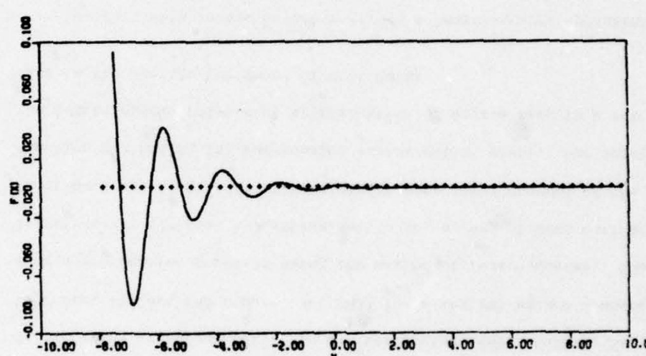



FIGURE 3.3c

Cubic Spline Regression: (magnified)



The term "local dependence" was originally applied to least-squares spline approximation by Powell [P2] in a paper bounding the effect of local perturbations of the approximated function. Subsequently, a number of other authors have developed results in three basic areas: 1) simple local dependence, in which the effect of local perturbations is bounded, 2) local convergence, in which the local error in least-squares spline approximation is bounded in terms of local properties of the knots and the approximated function, and 3) local solution, in which the least-squares spline is computed independent of distant knots and data.

Early local dependence bounds for the spline interpolate (which can be viewed as a special case of the least-squares spline) were derived by Ahlberg, Nilson, and Walsh [A2] for uniform knots and arbitrary-order splines, and by Kershaw [K3] for nonuniform knots and cubic splines. Later, Kammerer, Reddien, and Varga [K1,K2] employed the Kershaw result to derive local convergence bounds for the cubic and quadratic spline interpolates. Other local dependence results for spline interpolates were proved by Schoenberg [S2] while studying the asymptotic behavior of the cardinal spline basis functions.

Douglas, DuPont and Wahlbin [D8,D9], deBoor [B6], and Demko [D3,D4] have derived simple local dependence bounds for least-squares splines and have used these bounds to develop local, $L_\infty$ error bounds for least-squares splines. Eisenstat, Lewis, and Schultz [E2,E4] have applied local solution algorithms in computing solutions to large least-squares splines problems in limited storage.



In all of this work, the local dependence properties of spline approximations were derived from the corresponding local dependence properties of the associated linear systems. The early results were based on the explicit computation of the inverses of certain special matrices (e.g., tridiagonal matrices [K3,M2]). More recently some of these results have been extended to more general classes of matrices by Domsta [D7], Demko [D3,D4], and Hager and Strang [H1].

In Chapter IV, a unified theory of local dependence is developed for symmetric, positive definite, block tridiagonal matrices and the associated linear systems. The results include bounds on elements of the inverse, simple local dependence bounds, new error bounds for local solutions to linear systems, new error bounds for local inverses of matrices, and new error bounds for local Cholesky factorizations of matrices.

The least-squares spline local dependence theory of Chapter V follows easily from this matrix local dependence theory, The results include simple local dependence bounds, new error bounds for local least-squares spline approximations, and improved local $L_\infty$ error bounds for least-squares spline approximation. By applying the discrete least-squares stability analysis of §II.5, the results are extended to spline regression (least-squares approximation of data) and to more general classes of splines (such as L-splines [S6]).

## 1.4 Algorithms

Least-squares splines can be computed more efficiently than many classical approximations. When written in terms of the B-spline basis [C3,B3,B4], least-squares spline algorithms are both stable and efficient. In Chapter III, a detailed analysis of B-spline algorithms for computing and evaluating least-squares spline approximations to data is presented. The algorithms are given explicitly in an ALGOL-like language (see Appendix A) and operation counts are presented. Of particular interest are a fast, exact, incremental algorithm for evaluating splines (§III.5), a local algorithm for computing piecewise polynomial representations of splines (§III.6), and a local algorithm for computing piecewise polynomial representations of B-splines (§III.7).

In Chapter VI, the algorithm analysis of Chapter III and the local solution schemes of Chapter V are combined in creating a highly efficient, limited-storage algorithm for least-squares spline data fitting (see Table 4.1). The algorithm is intended for data fitting and signal processing applications where large quantities of data are processed on-line in small computers. The algorithm scans the data only once, producing the B-spline coefficients for the least-squares spline as the data are scanned. For a fixed relative accuracy, the algorithm requires a fixed amount of storage (a few hundred locations for cubic splines and $10^{-6}$ relative accuracy) and a fixed number of operations per data point (4 to 12 multiplications for cubic splines).



TABLE 4.1

Storage and Operation Counts for Cubic Spline Regression with n-2 Knots (§VI.5) and Polynomial Regression of Order n [D1], Both for N Equally Spaced Data Points

| | Storage | Operations |
|---|---|---|
| Spline | ~100 | 4N |
| Polynomial | ~n | ~3Nn |

## 1.5 Remarks

Many authors consider the principal advantage of spline approximation to be the ability to choose the knots [R1,p.123;R2;D6;P1]. Except for a local convergence result in §V.4, we neglect this topic altogether. Optimal knot placement generally involves nonlinear algorithms or heuristics which are significantly less efficient than linear least-squares algorithms. In many (but certainly not all) applications, optimal knot placement is not required to achieve an acceptable fit (see Figure VI.5.1) and the additional computational cost is not justified. However, in applications where optimal knot placement is important, the local solution algorithms of Chapter V and Chapter VI can provide significant speed-up by enabling the separate solution of small parts of the least-squares problem whenever knots are changed.

This dissertation is divided into six chapters: I) Introduction, II) Spline Approximation Theory, III) Algorithms, IV) Matrix Local Dependence, V) Least-Squares Spline Local Dependence, and VI) Real-Time Algorithm. A good overall picture of the work and its implications can be obtained by reading the introductions to the chapters and the numerical examples of §V.5. Open problems and future avenues for research are found in §IV.8 and §V.6.

While this dissertation treats only least-squares spline approximation, much of the work applies to other problems. The algorithm analysis of Chapter III can be extended easily to the Rayleigh-Ritz-Galerkin solution of elliptic partial differential equations; the *least-squares spline local dependence* results can be applied nearly verbatim to cubic spline interpolation, and the matrix local dependence results can be applied to 2-cyclic matrices.

# Chapter II

# Splines and Least-Squares Splines

## II.1 Introduction

The study of spline regression depends on a number of results in spline approximation theory. In this chapter, these results are introduced and the accompanying notation is defined.

The definitions of splines and the "B-spline" basis for splines [C3,B9] are introduced in §2. The B-spline basis is shown to be local and well-conditioned. In §3, the least-squares spline approximation to a function is defined, and conditions for existence and uniqueness are derived. The normal equations are also shown to be well-conditioned. In §4 and §5, all of these results are extended to the least-squares spline approximation of data.

In general, standard notation is employed. The notation pertaining to splines is similar to that of de Boor [B3] and Schultz [S5]. To aid in the interpretation of symbols, a list of notation is provided on pages v and vi.

## II.2 Splines and B-Splines

Like the infinitely thin draftman's splines (see Chapter I), spline functions are piecewise polynomials joined together with fixed continuity. Given a fixed order k (or degree k-1), a knot (or breakpoint) vector

(2.1) $\underline{u} \equiv (u_0, u_1, \ldots, u_{m+1}), \quad u_0 < u_1 < \ldots < u_{m+1}, \quad m > 0,$

and an incidence (or multiplicity) vector

(2.2) $\underline{z} \equiv (z_1, \ldots, z_m), \quad 1 \le z_i \le k, \quad 1 \le i \le m,$

a function s(t) in the collection of splines $S(k,\underline{u},\underline{z})$ is a polynomial of degree k-1 in each interval $(u_i, u_{i+1})$, $0 \le i \le m$, and at each knot $u_i$, $1 \le i \le m$, the spline has at least $k-1-z_i$ continuous derivatives. For example, the infinitely thin draftsman's splines are the set $S(4,\underline{u},\underline{1})$, i.e., the set of piecewise cubic polynomials having continuous curvature.

Splines can also be defined in terms of a "B-spline" (or basis-spline [C3]) expansion. Given a fixed order k and a B-spline knot vector

(2.3) $\underline{t} \equiv (t_1, t_2, \ldots, t_{n+k})$

satisfying the monotonicity restrictions

(2.4) $t_1 \le t_2 \le \ldots \le t_{n+k}$ and $t_i < t_{i+k}, \quad 1 \le i \le n,$



a function s(t) in the collection of splines $S(k,\underline{t})$ is a linear combination

(2.5) $s(t) = \sum_{i=1}^{n} a_i N_{i,k}(t), \quad \underline{a}$ real,

of the normalized B-spline functions (see Figure 2.1 and [C3,B4]). The B-splines are given by

(2.6) $N_{i,k}(t) \equiv (t_{i+k}-t_i)\, g_k^+(t_i, \ldots, t_{i+k};t), \quad 1 \le i \le n,$

where $g_k^+(t_i, \ldots, t_{i+k};t)$ is the $k^{th}$ divided difference in v for fixed t of the truncated power function

$$g_k^+(v;t) \equiv \begin{cases} (v-t)^{k-1} & \text{for } v \ge t \\ 0 & \text{otherwise} \end{cases}.$$

The B-spline definition (2.6) is not particularly suitable for practical computation. If several knots are coincident, then the divided difference has meaning only in the limit; and if several knots are nearly coincident, then the divided difference is numerically unstable. A more satisfactory (and stable [C2]) expression for the B-splines is the recurrence relation [B3,C2]

$$N_{i,1}(t) = \begin{cases} 1, & \text{if } t_i \le t < t_{i+1} \text{ or } t = t_{i+1} = t_{n+1} \\ 0, & \text{otherwise} \end{cases}, \quad 1 \le i \le n+k-1,$$

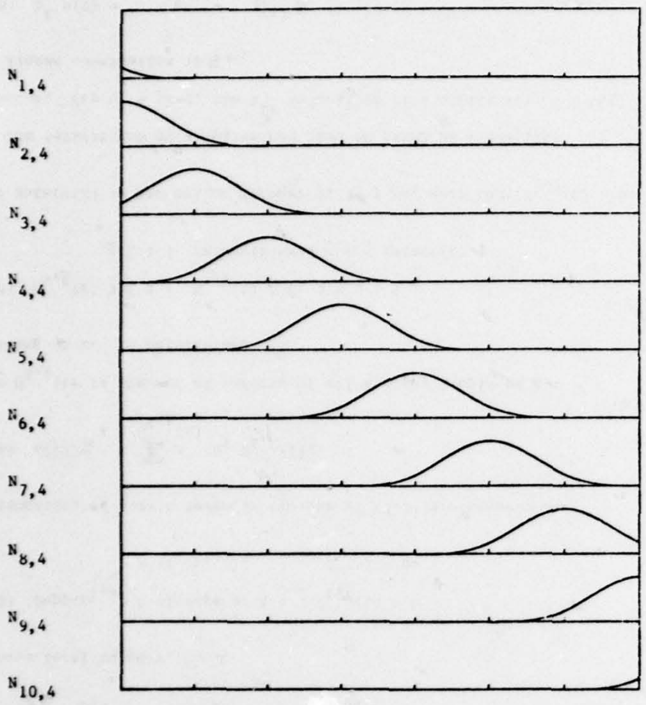
(2.7)

$$N_{i,r}(t) = (t-t_i)\frac{N_{i,r-1}(t)}{t_{i+r-1}-t_i} + (t_{i+r}-t)\frac{N_{i+1,r-1}(t)}{t_{i+r}-t_{i+1}}, \quad 1 \le i \le n, \ r \ge 2,$$

where the quantity $\frac{0}{0}$ is taken to be 0.

FIGURE 2.1

The B-splines for $k = 4$, $n = 10$, and
$\underline{t} = ( -3,-2,-1,0, 1, 2, 3, 4, 5, 6, 7,8,9,10 )$






For any collection of splines $S(k,\underline{u},\underline{z})$, if

$$(2.8) \qquad n \equiv k + \sum_{i=1}^{m} z_i,$$

and the B-spline knot vector $\underline{t}$ satisfies

$$(2.9) \qquad \begin{aligned} t_1 \le t_2 \le \cdots \le t_k &= u_0 \\ t_{k+1} = \cdots = t_{k+z_1} &= u_1 \\ t_{k+z_1+1} = \cdots = t_{k+z_1+z_2} &= u_2 \\ &\vdots \\ u_{m+1} = t_{n+1} \le t_{n+2} \le \cdots &\le t_{n+k}, \end{aligned}$$

then the B-splines are a basis for $S(k,\underline{u},\underline{z})$ [C3,B5]. Thus, for $\underline{t}$, $\underline{u}$, and $\underline{z}$ satisfying (2.9), the two sets $S(k,\underline{t})$ and $S(k,\underline{u},\underline{z})$ are equal and the two spline representations are equivalent.

Note that the incidence vector $\underline{z}$ gives the number of times that each knot in $\underline{u}$ occurs in the B-spline knot vector $\underline{t}$, i.e., the multiplicity of the knots of $\underline{u}$. Moreover, except for the monotonicity conditions (2.4), the values of the end knots $t_1,\ldots,t_{k-1}$, $t_{n+2},\ldots,t_{n+k}$ are unrestricted and can be chosen for notational or computational convenience. Generally we choose $t_1 = \cdots = t_k$ and $t_{n+1} = \cdots = t_{n+k}$ so that all n B-splines vanish outside of the interval $[t_1,t_{n+k}]$.

The B-splines are an especially advantageous representation for splines. It can be shown [B9] that the B-splines are non-negative, i.e.,

$$(2.10) \quad N_{i,k}(t) \ge 0 \quad \text{for all } t,\ 1 \le i \le n;$$

sum to unity, i.e.,

(2.11) $\sum_{i=1}^{N} N_{i,k}(t) = 1, \qquad t_k \le t \le t_{n+1};$

integrate to $(t_{i+k}-t_i)/k$, i.e.,

(2.12) $\int_{-\infty}^{+\infty} N_{i,k}(t) = \frac{t_{i+k}-t_i}{k}, \qquad 1 \le i \le n;$

and have local support, i.e.,

(2.13) $\mathrm{supp}(N_{i,k}) \equiv \text{closure of } \{ t \mid N_{i,k}(t) > 0 \}$

$= [t_i, t_{i+k}], \qquad 1 \le i \le n.$

Consequently, at most k terms in the sum of (2.5) are nonzero and

(2.14) $s(t) = \sum_{i \in M_{k,\underline{t}}(t)} a_i N_{i,k}(t),$

where $M_{k,\underline{t}}(t)$ is the set of indices of all k-order B-splines not vanishing at t. In particular,

(2.15) $M_{k,\underline{t}}(t) \equiv \{ i \mid N_{i,k}(t) > 0,\ 1 \le i \le n \}$

$\subseteq \{ i \mid \mathrm{interv}(t)-k+1 \le i \le \mathrm{interv}(t) \}$

where interv(t) is the unique integer j, $1 \le j \le n$, such that $N_{j,1}(t) > 0$.

The derivatives of a spline can also be given as a B-spline expansion. For $0 \le \ell \le k-1$, the $\ell^{th}$ derivative of a spline $s(t) \in S(k,\underline{t})$ is a linear combination [B3]

(2.16) $D^{\ell} s(t) = \sum_{i \in M_{k-\ell,\underline{t}}(t)} a_i^{(\ell)} N_{i,k-\ell}(t)\ ,$



of $k-\ell$ order B-splines, where

$a_i^{(0)} \equiv a_i, \quad 1 \le i \le n$

(2.17)

$a_i^{(\ell)} \equiv (k-\ell)\, \frac{a_i^{(\ell-1)} - a_{i-1}^{(\ell-1)}}{t_{i+k-\ell} - t_i},$

$\ell+1 \le i \le n, \quad t_{i+k-\ell}-t_i > 0, \quad 1 \le \ell \le k-1.$

Note that $a_i^{(\ell)}$ is not defined for all values of i and $\ell$, since some of the elements $a_i^{(\ell)}$ will never appear in the sum (2.16).

Another important property of the B-spline basis is that a weighted $\ell_p$ norm of the basis coefficients is equivalent to the $L_p$ norm of the corresponding spline, i.e., the basis coefficients are roughly the same "size" as the spline.

<u>THEOREM 2.1</u> [B2,B5,B7,B8]

There exists a positive constant $\Lambda_k$ independent of $\underline{t}$ such that

(2.18) $\Lambda_k^{-1} \| E^{1/p}\underline{a} \|_{\ell_p} \le \| \sum_{i=1}^{n} a_i N_{i,k} \|_{L_p} \le \| E^{1/p}\underline{a} \|_{\ell_p},$

$\underline{a}$ real, $\quad 1 \le p \le \infty$,

where

$E \equiv \mathrm{diag}(e_1, e_2, \ldots, e_n),$

(2.19)

$e_i \equiv \| N_{i,k} \|_{L_1} = \frac{t_{i+k}-t_i}{k}, \qquad 1 \le i \le n;$

and

$$(2.20)\quad \tfrac{1}{2}(\tfrac{\pi}{2})^k \le \Lambda_k \le 2k\, 9^{k-1}, \quad k \ge 1.$$

In particular, $\Lambda_1 = 1.$, $\Lambda_2 \le 2.5$, $\Lambda_3 \le 5.3$, and $\Lambda_4 \le 10.1$.

An immediate consequence of Theorem 2.1 is a bound on the E-scaled p-condition number of the B-spline basis

$$(2.21)\quad \kappa_p^E(\{N_{i,k}\}) \equiv \frac{\max\limits_{\|E^{1/p}\underline{a}\|_{\ell_p}=1} \|\sum_{i=1}^{n} a_i N_{i,k}\|_{L_p}}{\min\limits_{\|E^{1/p}\underline{a}\|_{\ell_p}=1} \|\sum_{i=1}^{n} a_i N_{i,k}\|_{L_p}} .$$

We can show that the B-spline basis is well-conditioned for any choice of $\underline{t}$ satisfying the monotonicity restrictions (2.4).

<u>COROLLARY 2.2</u>

If the knot vector $\underline{t}$ satisfies (2.4), then

$$(2.22)\quad \kappa_p^E(\{N_{i,k}\}) \le \Lambda_k \le 2k\, 9^{k-1}, \quad k \ge 1, \quad 1 \le p \le \infty.$$



## II.3 <u>Least-Squares Splines</u>

For any function $f(t) \in L_2[t_k, t_{n+1}]$, the least-squares spline projection $P_{S(k,\underline{t})}f(t)$ onto the space $S(k,\underline{t})$ is the unique spline $s(t) \in S(k,\underline{t})$ which minimizes the squared $L_2$-norm of the error

$$(3.1)\quad \|\epsilon(t)\|^2_{L_2[t_k,\, t_{n+1}]} = (\epsilon, \epsilon)_{L_2}, \qquad \epsilon(t) \equiv s(t) - f(t),$$

where

$$(p, q)_{L_2} \equiv \int_{t_k}^{t_{n+1}} p(t)q(t)\,dt.$$

If the spline $s(t)$ is written as the linear combination of B-splines (2.5), then the squared norm of the error (3.1) can be written as the quadratic form

$$(3.2)\quad \underline{a}^T G\, \underline{a} + 2\underline{a}^T \underline{b} + (f, f)_{L_2},$$

where the Gram matrix G (or Gramian) is given by

$$(3.3)\quad G \equiv [\, g_{i,j}\, ]_{n\times n}, \quad g_{i,j} \equiv (N_{i,k}, N_{j,k})_{L_2},$$

and the vector $\underline{b}$ is given by

$$(3.4)\quad \underline{b} \equiv [\, b_i\, ]_n, \qquad b_i \equiv (N_{i,k}, f)_{L_2},$$

The quadratic form (3.2) has a unique minimum at some $\underline{a}^f$ if and only if the matrix G is positive definite and the vector $\underline{a}^f$ satisfies the normal equations [S8,p.220]

$$(3.5)\quad G\, \underline{a}^f = \underline{b}.$$

Since the B-splines are a basis for $S(k,\underline{t})$, the Gram matrix is positive definite [C1,p.103], and the least-squares spline is unique.

The linear system (3.5) is particularly easy to solve. The Gramian G is a symmetric, nonnegative, positive definite matrix with bandwidth 2k-1 (e.g., Figure 3.1 and Appendix B). Moreover, as we show in the following corollary, the E-scaled 2-condition number of the Gramian

$$(3.6)\quad \kappa_2^E(G) \equiv \frac{\max\limits_{\|E^{1/2}\underline{a}\|_2 = 1} \underline{a}^T G\,\underline{a}}{\min\limits_{\|E^{1/2}\underline{a}\|_2 = 1} \underline{a}^T G\,\underline{a}}$$

is bounded independent of $\underline{t}$, so that the rounding error in solving the normal equations by Gaussian elimination is also bounded [F2,§20].

<u>COROLLARY 3.1</u>

If the knot vector $\underline{t}$ satisfies (2.4), then

$$(3.7)\quad \kappa_2^E(G) = \kappa_2^E(k,\underline{t})^2 \leq 4k^2\, 81^{k-1}.$$

<u>Proof:</u> From (3.3),

$$\begin{aligned} \underline{a}^T G\,\underline{a} &= \sum_{i=1}^{n}\sum_{j=1}^{n} \left( a_i N_{i,k},\ a_j N_{j,k} \right)_{L_2} \\ &= \left( \sum_{i=1}^{n} a_i N_{i,k},\ \sum_{j=1}^{n} a_j N_{j,k} \right)_{L_2} \\ &= \Big\| \sum_{i=1}^{n} a_i N_{i,k} \Big\|_{L_2}^2 . \end{aligned}$$

The result follows from (3.6), (2.21), and Corollary 2.2.

Q.E.D.



<u>FIGURE 3.1a</u>

The B-Spline Gram Matrix
k = 1 and knot spacing h

$$G = h \begin{bmatrix} 1 & & \bigcirc \\ & \ddots & \\ \bigcirc & & 1 \end{bmatrix}$$

<u>FIGURE 3.1b</u>

The B-Spline Gram Matrix
k = 2 and knot spacing h

$$G = \frac{h^2}{3!} \begin{bmatrix} 2 & 1 & & & \\ 1 & 4 & 1 & & \bigcirc \\ & \ddots & \ddots & \ddots & \\ \bigcirc & & 1 & 4 & 1 \\ & & & 1 & 2 \end{bmatrix}$$

<u>FIGURE 3.1c</u>

The B-Spline Gram Matrix
k = 3 and knot spacing h

$$G = \frac{h^3}{5!} \begin{bmatrix} 24 & 14 & 2 & & & & & & \\ 14 & 40 & 25 & 1 & & & & & \\ 2 & 25 & 66 & 26 & 1 & & & \bigcirc & \\ & 1 & 26 & 66 & 26 & 1 & & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & & 1 & 26 & 66 & 26 & 1 & \\ & \bigcirc & & & 1 & 26 & 66 & 25 & 2 \\ & & & & & 1 & 25 & 40 & 14 \\ & & & & & & 2 & 14 & 24 \end{bmatrix}$$

In practice, the bound of Corollary 3.1 is somewhat pessimistic, but it does reflect the exponential growth of the condition number with increasing k. For example, with infinitely many uniformly spaced knots, the Gram matrix is a bi-infinite Toeplitz matrix and its eigenvalues can

be computed explicitly (see Appendix B and Table 3.1). The condition number increases approximately as

(3.8) $\frac{1}{2}(\frac{\pi}{2})^{2k} = .5 \times 2.46740^k$ ,

which is far smaller than the upper bound of (3.7), but still exponential in k.

TABLE 3.1

The $\ell_2$ Condition Number of the B-Spline Gram Matrix for Infinitely Many Uniformly Spaced Knots

| Order | Condition Number |
|---|---|
| 1 | $\frac{1}{1}$ = 1.000 |
| 2 | $\frac{3}{1}$ = 3.000 |
| 3 | $\frac{15}{2}$ = 7.500 |
| 4 | $\frac{315}{17}$ = 18.529 |
| 5 | $\frac{2835}{62}$ = 45.726 |
| 6 | $\frac{155925}{1382}$ = 112.826 |
| 7 | $\frac{6081075}{21844}$ = 278.386 |

## II.4 Discrete Least-Squares Splines

The development for the least-squares spline approximation of data is similar to the preceding development for the approximation of functions. For a set of weighted data

(4.1) $Y \equiv \{ (x_\ell, w_\ell, y_\ell) \mid w_\ell > 0,\ x_\ell \in [t_k, t_{n+1}],\ 1 \le \ell \le N \}$,



the discrete least-squares spline projection onto the space $S(k,\underline{t})$ is the unique spline $s(t) \in S(k,\underline{t})$ which minimizes the squared $\ell_2^w$ norm of the error

$$\| \underline{\epsilon} \|^2_{\ell_2^w} = ( \underline{\epsilon}, \underline{\epsilon} )_{\ell_2^w} ,$$

(4.2)

$$\underline{\epsilon} \equiv ( \epsilon_1, \epsilon_2, \ldots, \epsilon_N ), \quad \epsilon_\ell = y_\ell - s(x_\ell), \quad 1 \le \ell \le N,$$

where the weighted $\ell_2$ inner product is defined as

(4.3) $( \underline{p}, \underline{q} )_{\ell_2^w} \equiv \sum_{\ell=1}^{N} w_\ell p_\ell q_\ell$ .

In data-fitting applications, the weights $w_\ell$ are frequently chosen as [B1]

$$w_\ell = (\sigma_\ell)^{-2}, \quad 1 \le \ell \le N,$$

where $\sigma_\ell$ is the uncertainty in measurement of the $\ell^{th}$ *data point.*

As in § 3, the basis coefficient vector $\underline{a}^Y$ of the $\ell_2^w$ spline projection satisfies the normal equations

(4.4) $\tilde{G}\, \underline{a}^Y = \tilde{\underline{b}}$ ,

where the discrete Gram matrix $\tilde{G}$ and the vector $\tilde{\underline{b}}$ are defined as in (3.3-3.4) using the $\ell_2^w$ inner product (4.3). If the matrix $\tilde{G}$ is positive definite, then a unique solution exists [S8,p.220].

However, the discrete Gram matrix may not be positive definite for all sets of data. For example, consider a set of data with $N >> n$ points, all of which lie in the support of $N_{1,k}$. In this case, the $(k+1)^{th}$ and following rows of the Gram matrix are identically zero.

Clearly the discrete Gram matrix is singular, and the discrete least-squares spline is not unique.

Before deriving sufficient conditions for uniqueness in general spline regression, we will consider the special case of data such that $N = n$, i.e., data for which the least-squares spline is a generalized spline interpolate. The conditions for existence and uniqueness of such an interpolate have been determined by Schoenberg and Whitney [S3] and others [R1,§7; B9,§7].

__THEOREM 4.1__ [S3;R1,§7;B9,§7]

For the set of n points

(4.5) $V \equiv \{ v_1 < v_2 < \dots < v_n \}$,

there exists a unique spline $s(t) \in S(k,\underline{t})$ satisfying

(4.6) $s(v_i) = y_i, \quad y_i \text{ real}, \quad 1 \le i \le n,$

if and only if

(4.7) $N_{i,k}(v_i) > 0, \quad 1 \le i \le n.$

Given this result, a sufficient condition for general spline regression follows immediately (cf. [S5,§6]). (The converse is also true, but the proof is somewhat lengthy [E5].)



__THEOREM 4.2__

If there exists a set

(4.8) $V \equiv \{ v_1 < v_2 < \dots < v_n \} \subseteq X$

satisfying (4.7), then the discrete Gram matrix is positive-definite and the least-squares spline is unique. Otherwise, the discrete Gram matrix is nonnegative-definite and there are infinitely many splines which minimize (4.2).

__Proof:__ If $s(t) \in S(k,\underline{t})$ is expressed as a B-spline expansion (2.5), then

$$\underline{a}^T \tilde{G}\, \underline{a} = ( s, s )^2_{\ell^w_2}$$

$$= \sum_{\ell=1}^{N} w_\ell\, s(x_\ell)^2$$

$$\ge \{ \min_{1 \le \ell \le N} w_\ell \} \{ \sum_{i=1}^{n} s(v_i)^2 \} \ge 0$$

so that the discrete Gram matrix is nonnegative definite and the normal equations (4.4) have at least one solution [S8,220].

If there exists a set V X satisfying the hypotheses, then from Theorem 4.1, the last sum is zero if and only if the spline is identically zero. Since the B-splines are a basis for $S(k,\underline{t})$, the spline s(t) is identically zero if and only if the basis coefficient vector $\underline{a}$ is zero. Thus, the Gram matrix is positive definite, the normal equations have a unique solution, and the discrete least-squares spline is unique [S8,220].

Q.E.D.

## II.5 The $X_p$ Norm

If the abscissa vector $\underline{x}$ satisfies the hypotheses of Theorem 4.2, then for all $s(t) \in S(k,\underline{t})$,

$$s(x_\ell) = 0, \quad 1 \le \ell \le N, \quad \underline{\text{if and only if}} \quad s(t) \equiv 0.$$

Consequently, the "discrete $L_p$" seminorms

$$\| s \|_{X_p} \equiv \Big( \sum_{\ell=1}^{N} w_\ell \, |s(x_\ell)|^p \Big)^{1/p}, \quad 1 \le p < \infty$$

$$\| s \|_{X_\infty} \equiv \max_{1 \le \ell \le N} |s(x_\ell)|$$

are norms on $S(k,\underline{t})$. In this section, with some restrictions on the data and knots, we prove the analog of Theorem 2.1 for any $X_p$ norm, i.e., we show that any $X_p$ norm over splines is equivalent to a weighted $\ell_p$ norm over the corresponding B-spline basis coefficients. Consequently, the B-spline basis is well-conditioned in the $X_p$ norm, the $X_p$ norm over splines is equivalent to the $L_p$ norm, and the discrete Gram matrix is well-conditioned.

As in §2, the scaling matrix $\tilde{E}$ is chosen to be

$$\tilde{E} \equiv \mathrm{diag}( \tilde{e}_1, \tilde{e}_2, \ldots, \tilde{e}_n ),$$

(5.1)

$$\tilde{e}_i \equiv \| N_{i,k} \|_{X_1}, \quad 1 \le i \le n.$$

For this scaling, the right-hand inequality of Theorem 2.1 follows immediately.



**LEMMA 5.1**

For all $\underline{t}$, $\underline{x}$, and $\underline{w}$,

$$\| \sum_{i=1}^{n} a_i N_{i,k} \|_{X_p} \le \| \tilde{E}^{1/p} \underline{a} \|_{\ell_p}, \quad \underline{a} \text{ real.} \tag{5.2}$$

Proof: If $p = \infty$, then from (2.11)

$$\| \sum_{i=1}^{n} a_i N_{i,k} \|_{X_\infty} \le \| \underline{a} \|_{\ell_\infty} \| \sum_{i=1}^{n} N_{i,k} \|_{X_\infty}$$
$$\le \| \underline{a} \|_{\ell_\infty} .$$

Otherwise, if $1 \le p < \infty$, then for $\frac{1}{p} + \frac{1}{q} = 1$, we have

$$\| \sum_{i=1}^{n} a_i N_{i,k} \|_{X_p} = \| \sum_{i=1}^{n} ( |a_i| N_{i,k}^{1/p} ) ( N_{i,k}^{1/q} ) \|_{X_p} .$$

From the Holder inequality [C1,p.45],

$$\| \sum_{i=1}^{n} a_i N_{i,k} \|_{X_p}$$
$$\le \| \Big( \sum_{i=1}^{n} ( |a_i| N_{i,k}^{1/p} )^p \Big)^{1/p} \Big( \sum_{j=1}^{n} ( N_{j,k}^{1/q} )^q \Big)^{1/q} \|_{X_p}$$
$$= \| \Big( \sum_{i=1}^{n} |a_i|^p N_{i,k} \Big)^{1/p} \Big( \sum_{j=1}^{n} N_{j,k} \Big)^{1/q} \|_{X_p} .$$

Since the B-splines sum to unity (see (2.11)),

$$\| \sum_{i=1}^{n} a_i N_{i,k} \|_{X_p} \le \| \Big( \sum_{i=1}^{n} |a_i|^p N_{i,k} \Big)^{1/p} \|_{X_p}$$
$$= \Big( \sum_{\ell=1}^{N} w_\ell \sum_{i=1}^{n} |a_i|^p N_{i,k}(x_\ell) \Big)^{1/p} .$$

Reversing the order of summation

$$\| \sum_{i=1}^{n} a_i N_{i,k} \|_{X_p} \leq \left( \sum_{i=1}^{n} |a_i|^p \left[ \sum_{\ell=1}^{n} w_\ell N_{i,k}(x_\ell) \right] \right)^{1/p}$$

$$= \left( \sum_{i=1}^{n} |a_i|^p \; \| N_{i,k} \|_{X_1} \right)^{1/p}$$

$$\equiv \| \tilde{E}^{1/p} \underline{a} \|_{\ell_p} .$$

Q.E.D.

The left-hand inequality of Theorem 2.1 does not hold for the $X_p$ norm and unrestricted $\underline{t}$, $\underline{x}$, and $\underline{w}$; i.e., for $k \geq 2$, there is no constant $\Gamma_k$ such that

$$(5.3) \quad \Gamma_k^{-1} \| \tilde{E}^{1/p} \underline{a} \|_{\ell_p} \leq \| \sum_{i=1}^{n} a_i N_{i,k} \|_{X_p} \quad \textit{for all } \underline{t}, \underline{x}, \textit{ and } \underline{w}.$$

We will construct a counterexample for the piecewise-linear splines ($k = 2$) with [W1]

$\underline{t} = ( 1, 1, 2, 3, \ldots, n, n )$

$\underline{x} = ( 1, 2-\delta, 3-\delta, 4-\delta, \ldots, n-\delta ), \quad 0 \leq \delta < 1,$

$\underline{w} = \underline{1}$ .

Let $s^*(t)$ be the (unique) piecewise-linear spline satisfying

$$s^*(x_\ell) = (-1)^\ell, \quad 1 \leq \ell \leq n.$$



If $\delta = \frac{2}{3}$, then the basis coefficients of $s^*(t)$ satisfy the recurrence relation (see Figure 5.1)

$$a_1^* = -1$$

$$a_i^* = -2\, a_{i-1}^* - 3 \operatorname{sign}(a_{i-1}^*), \quad 2 \leq i \leq n.$$

Thus,

$$|a_i^*| = 2^{i+1} - 3, \quad 2 \leq i \leq n,$$

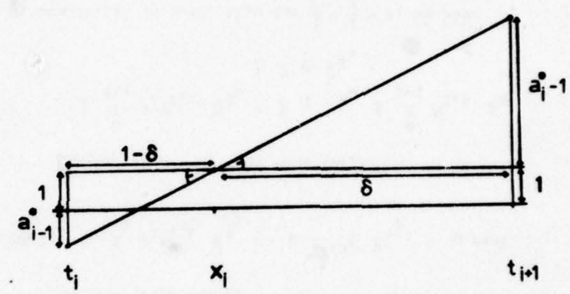
and

$$\| \underline{a}^* \|_{\ell_p} \geq 2^{n-1} .$$

The $\ell_p$ norm of the basis coefficient vector $\underline{a}^*$ grows exponentially with n, and there is no constant $\Gamma_2$ such that

$$\Gamma_2^{-1}\, 2^{n-1} \leq \Gamma_2^{-1} \| \underline{a}^* \|_{\ell_p} \leq \| s^* \|_{X_p} = n^{1/p}, \quad n \geq 1.$$

FIGURE 5.1
A Geometrical Construction for $a_i^*$



A norm equivalence relation analogous to Theorem 2.1 can be obtained if the knots, data, and weights are suitably restricted. Consider the class $Q_{k,\epsilon}$ of quasi-uniform $(\underline{t},\underline{x},\underline{w})$, i.e., for some positive $\epsilon \le \frac{1}{2}$, the set

$$(5.4)\quad Q_{k,\epsilon} \equiv Q_{k,\epsilon}(\Delta t_{min},\Delta t_{max},\Delta x_{min},w_{min},w_{max})$$
$$\equiv \{\ \begin{array}{lll} \Delta t_{min} \le t_{i+1}-t_i \le \Delta t_{max}, & 1\le i \le n+k-1; \\ \Delta x_{min} \le x_{\ell+1}-x_\ell \le \epsilon\,\Delta t_{min}, & 1\le \ell \le N-1; \\ w_{min} \le w_\ell \le w_{max}, & 1\le \ell \le N \end{array}\ \}.$$

Note that we are now restricted to smooth splines ($\underline{z} = 1$) and that N satisfies

$$\frac{t_{n+1}-t_k}{\epsilon\,\Delta t_{min}} \le N \le \frac{t_{n+1}-t_k}{\Delta x_{min}}.$$

In the following lemma, we use a compactness argument [cf. D5,F5] to derive a restricted norm equivalence relation. The constant obtained here depends on n.

LEMMA 5.2

If $n \ge 2k-1$ and $(\underline{t},\underline{w},\underline{x}) \in Q_{k,\epsilon}$, then there exists a positive constant $\Gamma_{k,n,\epsilon}$ such that

$$\Gamma_{k,n,\epsilon}^{-1}\,\|\tilde{E}^{1/p}\underline{a}\|_{\ell_p} \le \|\sum_{i=1}^{n} a_i N_{i,k}\|_{L_p}, \quad \underline{a} \text{ real}, \quad 1\le p \le\infty.$$

Proof: Consider the continuous real-valued function

$$f_k(\underline{a},\underline{t},\underline{x},\underline{w}) \equiv \|\sum_{i=1}^{n} a_i N_{i,k}\|_{X_p}.$$



Since $(\underline{t},\underline{x},\underline{w}) \in Q_{k,\epsilon}$ there will be at least 2k data points in each interval $[t_i,t_{i+k}]$, $1\le i \le n$. Consequently, the data satisfy the hypotheses of Theorem II.4.2 and the function $f_k(\underline{a},\underline{t},\underline{x},\underline{w})$ will be positive for all real $\underline{a} \ne 0$.

Because $\tilde{E} > 0$ for $(\underline{t},\underline{x},\underline{w}) \in Q_{k,\epsilon}$, the result is trivial for all $\underline{a} = 0$, and we can assume that

$$(5.5)\quad \|\tilde{E}^{1/p}\underline{a}\|_{\ell_p} = 1.$$

The set of $(\underline{a},\underline{t},\underline{x},\underline{w})$ satisfying (5.5) and $(\underline{t},\underline{x},\underline{w}) \in Q_{k,\epsilon}$ is a closed and bounded set. Consequently, the set is also compact and the function $f_k(\underline{a},\underline{t},\underline{x},\underline{w})$ achieves its infimum $\Gamma_{k,n,\epsilon}^{-1} > 0$ on that set [C1,§1.2].

Q.E.D.

To show that the constant $\Gamma_{k,n,\epsilon}^{-1}$ can be bounded away from zero, independent of n, we divide the interval $[t_k,t_{n+1}]$ into the subintervals $[t_k,t_{2k}]$, $[t_{2k},t_{3k}]$, ..., $[t_{n-k+1},t_{n+1}]$. (To simplify notation, we will assume that $n = Mk + k-1$ for some positive integer M.) We use Lemma 5.2 to show that the local $X_p$ norms

$$\|s\|_{X_p[t_{jk},t_{jk+k}]} \equiv \Big(\sum_{x_\ell \in [t_{jk},t_{jk+k}]} w_\ell\,|s(x_\ell)|^p\Big)^{1/p}, \quad 1\le j \le M,$$

are equivalent to the corresponding weighted local $\ell_p$ norms

$$\|\tilde{E}^{1/p}\underline{a}\|_{[jk-k+1,jk+k-1]} \equiv \Big(\sum_{i=jk-k+1}^{jk+k-1} \tilde{e}_i\,|a_i|^p\Big)^{1/p}, \quad 1\le j \le M.$$

Then we combine these local norm equivalence relations to derive the desired global norm equivalence relation.

THEOREM 5.3

If $(\underline{t},\underline{x},\underline{w}) \in Q_{k,\varepsilon}$, then

$$\Gamma^{-1}_{k,2k-1,\varepsilon} \|\tilde{E}^{1/p}\underline{a}\|_{\ell_p} \leq \|\sum_{i=1}^{n} a_i N_{i,k}\|_{X_p} \leq \|\tilde{E}^{1/p}\underline{a}\|_{\ell_p}, \quad \underline{a} \text{ real}, \ 1 \leq p \leq \infty.$$

Proof: The right-hand inequality follows from Lemma 5.1. If $1 \leq p < \infty$, then by Lemma 5.2,

$$\Gamma^{-p}_{k,2k-1,\varepsilon} \|\tilde{E}^{1/p}\underline{a}\|^p_{\ell_p[jk-k+1,jk+k-1]} \leq \|\sum_{i=1}^{n} a_i N_{i,k}\|^p_{X_p[t_{jk},t_{jk+k}]}, \quad 1 \leq j \leq M.$$

Summing the inequalities, we obtain

$$\Gamma^{-p}_{k,2k-1,\varepsilon} \sum_{j=1}^{M} \left( \sum_{i=jk-k+1}^{jk+k-1} \tilde{e}_i |a_i|^p \right) \leq \sum_{j=1}^{M} \|\sum_{i=1}^{n} a_i N_{i,k}\|^p_{X_p[t_{jk},t_{jk+k}]}$$

$$= \|\sum_{i=1}^{n} a_i N_{i,k}\|^p_{X_p[t_k,t_{n+1}]}.$$

Because the left-hand side is bounded below by

$$\|\tilde{E}^{1/p}\underline{a}\|^p_{\ell_p} = \sum_{i=1}^{n} \tilde{e}_i |a_i|^p \leq \sum_{j=1}^{M} \left( \sum_{i=jk-k+1}^{jk+k-1} \tilde{e}_i |a_i|^p \right)$$

the left-hand inequality of the result follows. The proof for $p = \infty$ is similar.

Q.E.D.

As in §2, this result leads to a bound on the $\tilde{E}$-scaled $X_p$ condition number of the B-spline basis and the 2-condition number of the $\tilde{E}$-scaled discrete Gram matrix, independent of the number of knots.



COROLLARY 5.4

If $(\underline{t},\underline{x},\underline{w}) \in Q_{k,\varepsilon}$, then

$$\kappa^{\tilde{E}}_{X_p}(\{N_{i,k}\}) \leq \Gamma_{k,2k-1,\varepsilon}, \quad 1 \leq p \leq \infty,$$

and

$$\kappa^{\tilde{E}}_{2}(\tilde{G}) \leq \Gamma^2_{k,2k-1,\varepsilon}.$$

The final corollary, which we will find useful in proving error bounds for spline regression, shows that the $L_p$ and $X_p$ norms are equivalent over $S(k,\underline{t})$.

COROLLARY 5.5

If $(\underline{t},\underline{x},\underline{w}) \in Q_{k,\varepsilon}$, then there exist positive constants $\lambda_Q$ and $\mu_Q$ independent of n such that

$$\lambda_Q \Gamma^{-1}_{k,2k-1,\varepsilon} \Lambda^{-1}_k \|s\|_{L_p} \leq \|s\|_{X_p} \leq \mu_Q \|s\|_{L_p}, \quad 1 \leq p \leq \infty.$$

Proof: Define

$$\lambda_Q \equiv \inf_{(\underline{t},\underline{x},\underline{w}) \in Q_k} \ \min_{1 \leq i \leq n} \left( \frac{\tilde{e}_i^{1/p}}{e_i^{1/p}} \right)$$

and

$$\mu_Q \equiv \sup_{(\underline{t},\underline{x},\underline{w}) \in \mathcal{Q}_k} \max_{1 \le i \le n} \left( \frac{\tilde{e}_i^{1/p}}{e_i^{1/p}} \right).$$

By a compactness argument similar to the proof of Lemma 5.2, we can show that the constants $\lambda_Q$ and $\mu_Q$ are positive and do not depend on N or n. Thus,

$$\lambda_Q \; \| \; E^{1/p}\underline{a} \; \|_{\ell_p} \le \| \; \tilde{E}^{1/p}\underline{a} \; \|_{\ell_p} \le \mu_Q \; \| \; E^{1/p}\underline{a} \; \|_{\ell_p}$$

and the result follows from the norm equivalence relations of Theorem 5.3 and Theorem 2.1.

Q.E.D.

# Chapter III

## Computing Least-squares Splines

### III.1 Introduction

In this chapter, several different algorithms for computing and evaluating least-squares splines are presented and operation counts are derived. Least-squares splines are computed by forming the normal equations for the B-spline basis [C3,B3] and solving the normal equations by an *envelope* $L\ D\ L^T$ *(or square-root-free Cholesky)* factorization algorithm [J1,G1,E1]. The resulting spline approximation can be evaluated as a B-spline expansion, or it can be converted to a piecewise polynomial for more efficient evaluation at numerous points.

The basic algorithm for forming the normal equations is developed in §2. The remaining sections are concerned with the efficient implementation of various subalgorithms: solving the normal equations in §3, locating intervals in §4, evaluating B-splines in §5, evaluating piecewise polynomials in §6, converting B-spline expansions to piecewise polynomials in §7, and forming the normal equations (faster) in §8.

### III.2 Forming the Normal Equations

From the modern numerical literature (e.g., [F1], [F2,§19], [D1,§5.7.1], [L1]), one might suppose that algorithms based on the normal equations would be unsuitable for any least-squares problem. Indeed, for many badly conditioned bases (such as the polynomials 1, $x^1$, $x^2$, $x^3$, ...), it is often impossible to obtain even a single digit of accuracy by solving the normal equations [F2,§19]. However, stable methods (e.g., QR decomposition [L1]) are neither necessary nor appropriate in computing low-order least-squares splines. The B-spline Gram matrix is well-conditioned* (see Corollary II.3.1 and Table II.3.2); the arithmetic required to solve the normal equations is about half that for any of the more stable methods [L1,§19]; and algorithms based on the normal equations are more compact.

Although the B-spline Gram matrix (II.3.3) is well-conditioned for low-order splines, independent of $\underline{t}$ (Corollary II.3.1), the discrete B-spline Gram matrix is not necessarily well-conditioned or even nonsingular for all sets of data (see §II.4). However, if a set of data satisfies the hypotheses of Theorem II.4.2, then the discrete Gram matrix is at least nonsingular. In addition, numerical experiments indicate that the $\tilde{E}$-scaled 2-condition number of the discrete Gram

---

* However, as the order of the spline increases, the condition number of the Gramian increases rapidly, approximately as $\frac{1}{2}(\frac{\pi}{2})^{2k}$ (see (II.3.8) and Table II.3.2). Numerical experiments indicate that the accuracy of the computed solutions drops rapidly. Thus, one should not contemplate using the normal equations to solve high-order (greater than degree 10 for $10^{-6}$ machine precision) least-squares spline problems.



matrices tends to follow the local mesh ratio (see Table 2.1)

$$\sigma \equiv \max_{1\le i \le n} \left( \frac{\| N_{i,k} \|_{X_1}}{N_{i+1,k} \|_{X_1}}, \frac{\| N_{i,k} \|_{X_1}}{N_{i+1,k} \|_{X_1}} \right),$$

but this rule is not infallible (e.g., the last two examples of Table 2.1).

TABLE 2.1

The $\ell_2$ Condition Number of Some $\tilde{E}$-scaled Discrete B-Spline Gram Matrices

| Knots and Data | Mesh Ratio | Condition Number |
|---|---|---|
| $k = 2$, $\underline{t}$ = (1, 1, 2, 3, 4, 5, 5) | | |
| $\underline{x}$ = (1, 2.00, 3, 4, 5) | 1.00 | 1.00 |
| $\underline{x}$ = (1, 1.10, 3, 4, 5) | 19.00 | 19.00 |
| $\underline{x}$ = (1, 1.01, 3, 4, 5) | 199.00 | 199.00 |
| $\underline{x}$ = (1, 1.00, 3, 4, 5) | ∞ | ∞ |
| $k = 4$, 41 uniformly spaced data points | | |
| $\underline{t}$ = (1, 1, 1, 1, 2.00, 3.00, 4.00, 5, 5, 5, 5) | 1.64 | 22.15 |
| $\underline{t}$ = (1, 1, 1, 1, 1.50, 4.00, 4.00, 5, 5, 5, 5) | 4.14 | 26.33 |
| $\underline{t}$ = (1, 1, 1, 1, 1.10, 4.50, 4.50, 5, 5, 5, 5) | 8.50 | 31.56 |
| $\underline{t}$ = (1, 1, 1, 1, 1.05, 4.75, 4.75, 5, 5, 5, 5) | 18.70 | 34.93 |
| $k = 1$, $\underline{t}$ = (1, 2, 10), $\underline{x}$ = (1, 2, 3, 4, 5, 6, 7, 8, 9) | 9.00 | 1.00 |
| $k = 4$, $\underline{t}$ = (1, 1, 1, 1, 2, 3, 4, 5, 5, 5, 5), $\underline{x}$ = (1, 2, 3, 4, 5) | 5.00 | ∞ |

The first task in solving a least-squares spline data-fitting problem is computing the discrete Gram matrix

$$G = [\, g_{i,j} \,]_{n\times n}, \quad g_{i,j} = \sum_{\ell=1}^{N} w_\ell \, N_{i,k}(x_\ell) N_{j,k}(x_\ell), \quad 1\le i,j \le n$$

and the vector

$$\underline{b} = [\, b_i \,]_n, \qquad b_i = \sum_{\ell=1}^{N} w_\ell \, N_{i,k}(x_\ell)\, y_\ell, \quad 1 \le i \le n.$$

Since the B-spline basis functions have local support, these sums can be written as

$$(2.1)\ g_{i,j} = \sum_{N_{i,k}(x_\ell) N_{j,k}(x_\ell) > 0} w_\ell \, N_{i,k}(x_\ell) N_{j,k}(x_\ell), \quad 1 \le i,j \le n,$$

and

$$(2.2)\ b_i = \sum_{N_{i,k}(x_\ell) > 0} w_\ell \, N_{i,k}(x_\ell) y_\ell, \quad 1 \le i \le n.$$

The Gram matrix is symmetric, so that we only need to compute its lower triangle.

Instead of applying (2.1) and (2.2) directly and computing the sums for each element of G and $\underline{b}$, we employ a data-directed approach. For each data point $x_\ell$, $1 \le \ell \le N$, we compute $i$ = interv($x_\ell$), the index of the interval of $\underline{t}$ containing $x_\ell$. Then we compute the k basis functions $N_{i-k+1,k}(x_\ell), \ldots, N_{i,k}(x_\ell)$ not vanishing trivially at $x_\ell$ and the corresponding terms of the sums (2.1) and (2.2).

The following algorithm is a rough sketch of the computations involved; in the remainder of this chapter we will develop the various subalgorithms in more detail. This algorithm requires approximately $2\,N\,k^2$ operations*. (The exact count depends on the choices for the various subalgorithms. See Table 8.2 for a summary.)

---

* Unless otherwise noted, the operations counted will be multiplications and divisions.



<u>ALGORITHM 2.1</u>: Forming the Normal Equations

Input:

| | |
|---|---|
| N | the number of data points |
| x[N] and y[N] | the data arrays |
| k | the order of the spline |
| n | the number of basis functions |
| t[n+k] | the knot vector |

Output:

| | |
|---|---|
| G[n,n] | the lower triangle of the discrete Gram matrix |
| b[n] | the right hand side |

<u>Algorithm</u>:

```
1   Zero  G and b
2   FOR ℓ=1 UNTIL N DO
2a       Compute the integer i = interval(x_ℓ)
2b       Evaluate the k B-splines  N_{i-k+1,k}(x_ℓ), ..., N_{i,k}(x_ℓ)
         not vanishing trivially at x_ℓ.
2c       Add the contribution of these k basis functions into G and b
         FOR r:=i-k+1 UNTIL i DO
           wn = N_{r,k}(x_ℓ)*w_ℓ
           b_r = b_r + wn*y_ℓ
           FOR s:=i-k+1 UNTIL r DO
             g_{r,s}:= g_{r,s} + wn*N_{s,k}(x_ℓ)
```

## III.3 <u>Storing and Solving the Normal Equations</u>

The basis coefficient vector of the least-squares spline is the solution to the normal equations

$$(3.1)\ G\,\underline{a} = \underline{b}.$$

Since the matrix G is symmetric, positive definite, and well-conditioned, an $L\,D\,L^T$ factorization algorithm will provide an accurate solution to this linear system [F2,§9,§23; M3]. The linear system is solved in three steps: computing L and D such that $G = L\,D\,L^T$ (the factorization), solving the triangular system $L\,\underline{c} = \underline{b}$ (the forward-solution), and solving the triangular system $D\,L^T\,\underline{a} = \underline{c}$ (the back-solution). We will consider the details of this algorithm later in this section; first, we consider schemes for storing the Gram matrix.

The Gram matrix G is zero except for a small band about the diagonal, i.e., in the lower triangle of G,

$$g_{i,j} \neq 0 \quad \underline{\text{if and only if}} \quad \text{there exists an } \ell \text{ such that } t_i < x_\ell < t_{j+k}\,.$$

For example, in the common special case of dense data with the knot multiplicity vector $\underline{z} = \underline{1}$, the discrete B-spline Gram matrix has bandwidth 2k-1 (see Figure 3.1); for coincident knots, the matrix has a staircase pattern (see Figure 3.1); and for sparse data, the matrix could have bandwidth as small as 2k-3 (see Figure 3.2).



FIGURE 3.1

The Non-Zero Structure of Cubic B-Spline Gram Matrices
(The symbol "X" represents a nonzero off-diagonal element
and the symbol "D" represents a nonzero diagonal element)

$\underline{z} = \underline{1}$

$$\begin{bmatrix}
D&X&X&X&&&&&&&&\\
X&D&X&X&X&&&&&\bigcirc&&\\
X&X&D&X&X&X&&&&&&\\
X&X&X&D&X&X&X&&&&&\\
&X&X&X&D&X&X&X&&&&\\
&&X&X&X&D&X&X&X&&&\\
&&&X&X&X&D&X&X&X&&\\
&&&&X&X&X&D&X&X&X&\\
&&&&&X&X&X&D&X&X&X\\
&\bigcirc&&&&&X&X&X&D&X&X\\
&&&&&&&X&X&X&D&X\\
&&&&&&&&X&X&X&D
\end{bmatrix}$$

$\underline{z} = \underline{2}$

$$\begin{bmatrix}
D&X&X&X&&&&&&&&\\
X&D&X&X&X&&&&&\bigcirc&&\\
X&X&D&X&X&&&&&&&\\
X&X&X&D&X&X&X&&&&&\\
&&X&X&D&X&X&&&&&\\
&&X&X&X&D&X&X&&&&\\
&&&&X&X&D&X&X&X&&\\
&&&&X&X&X&D&X&X&&\\
&&&&&&X&X&D&X&X&X\\
&\bigcirc&&&&&X&X&X&D&X&X\\
&&&&&&&&X&X&D&X\\
&&&&&&&&X&X&X&D
\end{bmatrix}$$

$\underline{z} = \underline{3}$

$$\begin{bmatrix}
D&X&X&X&&&&&&\\
X&D&X&X&&&&&\bigcirc&\\
X&X&D&X&&&&&&\\
X&X&X&D&X&X&X&&&\\
&&&X&D&X&X&&&\\
&&&X&X&D&X&&&\\
&&&X&X&X&D&X&X&X\\
&\bigcirc&&&&&X&D&X&X\\
&&&&&&X&X&D&X\\
&&&&&&X&X&X&D
\end{bmatrix}$$

$\underline{z} = \underline{4}$

$$\begin{bmatrix}
D&X&X&X&&&&&&&&\\
X&D&X&X&&&&&&\bigcirc&&\\
X&X&D&X&&&&&&&&\\
X&X&X&D&&&&&&&&\\
&&&&D&X&X&X&&&&\\
&&&&X&D&X&X&&&&\\
&&&&X&X&D&X&&&&\\
&&&&X&X&X&D&&&&\\
&&&&&&&&D&X&X&X\\
&\bigcirc&&&&&&&X&D&X&X\\
&&&&&&&&X&X&D&X\\
&&&&&&&&X&X&X&D
\end{bmatrix}$$

FIGURE 3.2

Discrete B-Spline Gram Matrices

$k = 2,\ \underline{t} = (0,1,2,3,4,5,6)$
$\underline{x} = (1,2,3,4,5)$

$$\begin{bmatrix}
1&&&&&\\
&1&&&\bigcirc&\\
&&1&&&\\
&&&1&&\\
\bigcirc&&&&1&\\
&&&&&1
\end{bmatrix}$$

$k = 4,\ \underline{t} = (0,1,2,3,4,5,6,7,8,9,10)$
$\underline{x} = (3,4,5,6,7)$

$$\frac{1}{36}\begin{bmatrix}
1&4&1&&&&&\\
4&17&8&1&&&\bigcirc&\\
1&8&18&8&1&&&\\
&1&8&18&8&1&&\\
&&1&8&18&8&1&\\
&&&1&8&18&8&1\\
&&&&1&8&17&4\\
\bigcirc&&&&&1&4&1
\end{bmatrix}$$

It is desirable to avoid storing, or operating on, the numerous zero entries in these matrices. One scheme which meets these goals is an envelope (or profile) storage and factorization algorithm [J1,G1,E1]. If the indices of the first nonzeros in each row of the lower triangle are given by

$$f_i \equiv \min\{\, j \mid g_{i,j} \neq 0,\ \ 1 \le j \le i \,\},\quad 1 \le i \le n,$$

then the symmetric envelope of the matrix G (or the envelope of the lower triangle of G) is the set of ordered pairs of indices

$$\mathrm{senv}(G) \equiv \{\, (i,j) \mid f_i \le j \le i,\ \ 1 \le i \le n \,\}.$$

For the B-spline Gram matrices, the symmetric envelope contains only the nonzeros of the lower triangle, *i.e.*,

$$\mathrm{senv}(G) = \{\, (i,j) \mid g_{i,j} \neq 0,\ 1 \le j \le i \le n \,\},$$

so that envelope algorithms are well-suited to least-squares spline computations.

The symmetric envelope of G can be stored in an array g[.] and the elements can be accessed as [E1]

$$g_{i,j} \equiv g[\, p[i]+j \,],\qquad f_i \le j \le i \le n,$$

where the pointer array p[.] is given by

$$p[1] \equiv 0$$

$$p[i] \equiv p[i-1] + i - f_i,\qquad 2 \le i \le n.$$



The array g[.] requires Q locations, where

$$Q \equiv n + p[n] = n + \sum_{i=1}^{n} (i-f_i).$$

For the spline space $S(k,\underline{u},\underline{z})$ and dense data, we can show from [G1,E1] that

$$Q = \frac{1}{2}\left( (m+1)k(k+1) - \sum_{i=1}^{m} (k-z_i)(k-z_i+1) \right).$$

The $L\,D\,L^T$ factorization of G and the solution to the linear system $G\,\underline{a} = \underline{b}$ can be computed by a variant of symmetric Gaussian elimination called envelope factorization [J1,G1,E1]. For the spline space $S(k,\underline{u},\underline{z})$ and dense data, we can show from [G1,E1] that factoring the matrix requires

$$\frac{1}{6}\left( (m+1)k(k-1)(k+4) - \sum_{i=1}^{m} (k-z_i)(k-z_i-1)(k-z_i+4) \right)$$

operations, and solving the triangular systems requires

$$2\,Q - n$$

operations. Furthermore, because

$$\mathrm{senv}(G) = \mathrm{senv}(L) \cup \mathrm{senv}(D),$$

the matrices D and L of the factorization can overwrite the matrix G in memory, so that no storage is needed beyond that required for G. Approximate operation counts and storage requirements for some special cases are given in Table 3.1. The index vector p[.] is not included in the storage counts.

TABLE 3.1

Storage and Operation Counts for Solving the Normal Equations

| z | Storage | Operation Counts | |
|---|---|---|---|
| | | Factorization | Solution |
| 1 | $\sim nk$ | $\frac{1}{2}nk(k+1)$ | $\sim n(2k-1)$ |
| k | $\frac{1}{2}(m+1)\,k(k+1)$ | $\frac{1}{6}(m+1)k(k-1)(k+4)$ | $(m+1)k(k+1)-n$ |

ALGORITHM 3.1: Solving the Normal Equations [J1,G1,E1].

Input:

| | |
|---|---|
| n | the number of unknowns |
| p[n] | the pointer array |
| g[n+p[n]] | storage vector for the Gram matrix |
| b[n] | the right hand side |

Output:

| | |
|---|---|
| g[n+p[n]] | the lower triangle of the L D $L^T$ factorization of G (written over G) |
| a[n] | the solution |

Algorithm:

```
COMMENT   Factorization

FOR ℓ:=1 UNTIL n DO
   fl := ℓ-p[ℓ]+p[ℓ-1]
COMMENT   Off diagonal elements
   FOR i:=fl UNTIL ℓ-1 DO
      FOR j:=max( i-p[i]+p[i-1], fl ) UNTIL i-1 DO
         g[p[ℓ]+i] := g[p[ℓ]+i] - g[p[i]+j] * g[j+p[ℓ]]
```



```
COMMENT   Diagonal elements and forward-solution
   a[ℓ] := b[ℓ]
   FOR i:=fl UNTIL ℓ-1 DO
      old := g[p[ℓ]+i]
      new := old / g[p[i]+i]
      g[p[ℓ]+i] := new
      g[p[ℓ]+ℓ] := g[p[ℓ]+ℓ] - new * old

      a[ℓ] := a[ℓ] - new * a[i]
COMMENT   Back-Solution

FOR ℓ:=n STEP -1 UNTIL 1 DO
   a[ℓ] := a[ℓ] / g[p[ℓ]+ℓ]

FOR ℓ:=n STEP -1 UNTIL 1 DO
   FOR i:=max( 1, ℓ-p[ℓ]+p[ℓ-1] ) UNTIL ℓ-1 DO
      a[i] := a[i] - g[i+p[ℓ]] * a[ℓ]
```

The linear system could also be solved by a band L D $L^T$ factorization algorithm [M3], which is similar to envelope factorization with the pointer array

$$p[1] = 0$$
$$p[i] = p[i-1] + \max(k-1,\ i-1), \qquad 2 \le i \le n.$$

In band factorization, the entire band of G is stored and factored, whether or not the elements are nonzero. A band factorization code is slightly simpler than an envelope code, and does not require use of a pointer array. (However, an envelope factorization algorithm may run considerably faster than a band factorization algorithm using a two-dimensional array to store the matrix, because many FORTRAN systems require an integer multiplication for each two-dimensional array access).

The band algorithm requires $\sim nk$ locations to store the matrix, $\sim\frac{1}{2}nk(k+1)$ operations to factor it, and $\sim n(2k-1)$ operations to solve the triangular systems. Thus, for $\underline{z} = 1$, the band and envelope schemes would require the same number of operations and approximately the same amount of storage; but, for $\underline{z} = k$, the envelope scheme would require one-third the operations to factor the matrix, one-half the storage, and one-half the operations to solve the triangular systems.

## III.4 Interval Location

Before evaluating a spline at a point t, whether in forming the normal equations or in evaluating the least-squares spline, we must compute interv(t), *the index of the* interval of the knot vector $\underline{t}$ which contains t. Computing interv(t) can be more expensive than evaluating the spline itself, particularly for large n, nonuniform knots, and randomly distributed evaluation points. In this section we develop several different algorithms for interval location, each for different assumptions on the distribution of the evaluation points and the knots.

More formally, given a vector $\underline{x}$ of N evaluation points, we wish to compute an integer vector $\underline{I}$ such that

$$I_\ell = \text{interv}(\ x_\ell\ ),\quad 1 \le \ell \le N.$$

For a knot vector with uniform spacing $h = t_{i+1} - t_i$, $k \le i \le n$, the problem is trivial. The following algorithm requires N divisions.



ALGORITHM 4.1: Uniform Knots

```
FOR ℓ:=1 UNTIL N DO
    I_ℓ := ⌊(x_ℓ - t_k)/h⌋ + k .
```

If the vector $\underline{x}$ is ordered, i.e., if $x_\ell \le x_{\ell+1}$ for $1 \le \ell \le N-1$, then the following simple algorithm suffices. This algorithm requires at most N+n floating point comparisons.

ALGORITHM 4.2: Ordered Data

```
i := 1
FOR ℓ:=1 UNTIL N DO
    WHILE( x_ℓ ≥ t_{i+1} AND i < n )
        i := i+1
    I_ℓ := i
```

If the knots are not uniformly spaced and the data abscissas are not ordered, then neither of the two simple algorithms can be employed. One possible alternative is to sort the vector $\underline{x}$, an operation which could require as many as $O(N \log_2 N)$ operations; and to employ Algorithm 4.2. If the ordering of the data points were significant, this approach would require additional storage for pointers or a second copy of the data. Furthermore, for large N, the cost of sorting the data could dominate the cost of forming ($O(Nk^2)$), see Table 8.2) and solving ($O(nk^2)$), see Table 3.1) the normal equations. Another approach is to locate each of the data points by binary search. This procedure

could require as many as $O(N \log_2 n)$ comparisons, still sufficiently many to dominate the cost of forming and solving the normal equations.

The ideal interval location scheme is an algorithm which is nearly as efficient as Algorithm 4.2 for sorted data, does not require rearrangement of the data, and works well for randomly distributed data. Furthermore, for data which are nearly sorted (i.e., for which the next abscissa is likely to be close to the previous abscissa), the algorithm should require average time proportional to N.

A two-phase, local binary search algorithm [B4] meets these requirements. In the first phase, larger and larger intervals around the previous data point are searched until an interval is found which contains the current data point. In the second phase, the intervals are halved successively until a single interval is found which contains the data point (i.e., binary search). For sorted data, the algorithm requires two floating point comparisons per data point; for randomly distributed data, the algorithm requires fewer than $2 N \log_2 n$ floating point comparisons. In general, the algorithm requires

$$\sum_{\ell=2}^{n} 2\left\lceil \log_2( 1 + |I_\ell - I_{\ell-1}| ) \right\rceil$$

comparisons. In particular, for data with K-bounded variation, i.e., for data such that

$$| I_\ell - I_{\ell-1}| \le K, \quad 2 \le \ell \le N,$$

the algorithm requires at most $2N \log_2( K )$ floating point comparisons.



<u>ALGORITHM 4.3</u>: Local Binary Search [B4]

```
Input:
        x[N]          the data vector
        k             the order of the spline
        n             the number of basis functions
        t[n+k]        the knot vector

Output:
        I[N]          the interval vector
```

<u>Algorithm</u>:

```
        low := 1
        FOR ℓ:=1 UNTIL N DO
          high := low+1
          width := 1

COMMENT    PHASE 1A: Expand search interval DOWN

          WHILE (  low > k  AND  x[ℓ] < t[low]  )  DO
            high := low
            low := low - width
            width = 2*width
          low := max( k, low )

COMMENT    PHASE 1B: Expand search interval UP

          WHILE (  high < n+1  AND  x[ℓ] ≥ t[high]  ) DO
            low := high
            high := high + width
            width := 2*width
          high := min( high, n+1 )

COMMENT    PHASE 2: Binary search

          WHILE  ( high-low > 1 )  DO
            mid := (low + high)/2
            IF  x[ℓ] < t[mid] THEN
              high := mid
            ELSE
              low := mid

          I[ℓ] := low
```

TABLE 4.1

Storage and Operation Counts
for Interval Location with N Data Points and n+k Knots
(operations are comparisons unless otherwise noted)

| Algorithm | Description | Operations (upper bound) |
|---|---|---|
| | binary search | $N \log_2 n$ |
| | sort | $N \log_2 N$ |
| 4.1 | Uniform knot spacing | N divisions |
| 4.2 | Sorted data | n+N |
| 4.3 | Local binary search | $2 N \log_2 n$ |
| | K-bounded variation | $2 N \log_2 K$ |

## III.5 The B-Spline Representation

Given an efficient scheme for interval location, the most time-consuming part of a least-squares spline calculation is either the evaluation of the basis functions in Step 2b of Algorithm 2.1 or the evaluation of the least-squares spline itself. In this section we consider schemes for evaluating a spline from its B-spline coefficients.



The most straightforward evaluation scheme is the sum (II.2.14)

$$(5.1) \quad s(t) = \sum_{j=i-k+1}^{i} a_j N_{j,k}(t), \qquad i = \mathrm{interv}(t).$$

To compute this sum, the k values $N_{i-k+1,k}(t), \ldots, N_{i,k}(t)$ are required. These values can be computed efficiently using the recurrence relation (II.2.7).

If $i = \mathrm{interv}(t)$, then the only basis function of order one not vanishing at t is $N_{i,1}(t) \equiv 1$. The two basis functions of order two not vanishing trivially at t are $N_{i-1,2}(t)$ and $N_{i,2}(t)$. Their values can be computed from (II.2.7) using the value of $N_{i,1}(t)$. Similarly, for orders $r = 3,\ldots,k$, the r basis functions $N_{i-r+1,r}(t), \ldots, N_{i,r}(t)$ not vanishing trivially at t can be computed from (II.2.7) using the r-1 previously computed values $N_{i-r+2,r-1}(t), \ldots, N_{i,r-1}(t)$. In the process the following k×k triangle of values is computed, column by column:

$$(5.2) \quad \begin{array}{cccc} N_{i,1}(t) & N_{i-1,2}(t) & \cdots & N_{i-k+1,k}(t) \\ 0 & N_{i,2}(t) & \cdots & N_{i-k+2,k}(t) \\ \cdot & \cdot & \cdot & \\ 0 & 0 & \cdots & N_{i-1,k}(t) \\ 0 & 0 & 0 & N_{i,k}(t) \end{array}$$

Computing the entire triangle of $\frac{1}{2}k(k+1)$ B-splines requires $\frac{3}{2}k(k-1)$ operations and $k(k+2)$ storage locations.

ALGORITHM 5.1: Evaluating the B-splines I [B3,B4]

Input:

| | |
|---|---|
| k | the order of the spline |
| x | the evaluation point |
| i | the integer interv( x ) |
| t[n+k] | the knot vector |

Output:

N[k,k] the values $N[j,r] = N_{i-r+j,r}(x)$, $1 \le j \le r$, $1 \le r \le k$.

Temporaries:
dp[k-1] dm[k-1]

Algorithm:

```
N[1,1] := 1
FOR r:=1 UNTIL k-1 DO
  dp[r] := t[i+r] - x
  dm[r] := x - t[i+1-r]
  N[1,r+1] := 0
  FOR s:= 1 UNTIL r DO
    vm := N[s,r]/(dp[s] + dm[r+1-s])
    N[s,r+1] := N[s,r+1] + dp[s]*vm
    N[s+1,r+1] := dm[r+1-s]*vm
```

If only the values in the last row of the table (5.2) are required, then without increasing the operation counts, the algorithm can be reorganized so that the storage requirement is reduced to 3k locations. Evaluating a spline using (5.1) and this B-spline algorithm requires $k + \frac{3}{2}k(k-1)$ operations. Evaluating another spline having the same knots at the same point requires an additional k operations.



ALGORITHM 5.2: Evaluating the B-splines II [B3,B4]

Input:

| | |
|---|---|
| k | the order of the spline |
| i | the evaluation point |
| t[n+k] | the knot vector |
| i | the integer interv( x ) |

Output:

v[k] the values $N_{i-k+1,k}(x), \ldots, N_{i,k}(t)$

Temporaries:
dp[k-1] dm[k-1]

Algorithm:

```
v[1] := 1
FOR r:=1 UNTIL k-1 DO
  dp[r] := t[i+r] - x
  dm[r] := x - t[i+1-r]
  vp := 0
  FOR s:= 1 UNTIL r DO
    vm := v[s]/(dp[s] + dm[r+1-s])
    v[s] := vp + dp[s]*vm
    vp := dm[r+1-s]*vm
  v[r+1]:= vp
```

The $\ell^{th}$ derivative of a spline, $0 \le \ell \le k-1$, is the $k-\ell$ order spline (II.2.16-17)

(5.3) $$D^{\ell} s(t) = \sum_{j=i-k+\ell+1}^{i} a_j^{(\ell)} N_{j,k-\ell}(t), \qquad i = \mathrm{interv}(t),$$

where

(5.4)
$$a_j^{(0)} \equiv a_j, \quad 1 \le j \le n,$$
$$a_j^{(\ell)} \equiv (k-\ell)\frac{a_j^{(\ell-1)} - a_{j-1}^{(\ell-1)}}{t_{j+k-\ell} - t_j},$$
$$\ell+1 \le j \le n, \quad t_{j+k-\ell} - t_j > 0, \quad 1 \le \ell \le k-1.$$

After the vector $\underline{a}^{(\ell)}$ has been computed, evaluating the $\ell^{th}$ derivative of a spline is a simple matter of evaluating the $k-\ell$ order spline (5.3).

Computing $\underline{a}^{(\ell)}$ requires n locations and fewer than $2n\ell$ operations; computing the $\ell^{th}$ derivative requires another 3k locations and $\frac{3}{2}(k-\ell)(k-\ell-1) + (k-\ell)$ operations.

For very little additional work, all k-1 nonvanishing derivatives can be evaluated at the same time as the spline. First, fewer than 2n(k-1) operations are required to compute the $\underline{a}^{(\ell)}$, $1 \leq \ell \leq k-1$. Then, for each evaluation of the spline and its k-1 derivatives, $\frac{3}{2}k(k-1)$ operations are required to compute the B-splines; and $\frac{1}{2}k(k+1)$ operations are required for the sums (5.3). As in many other computations involving the triangle of B-spline values (5.2), storage can be limited to 4k locations by integrating the derivative computation with Algorithm 5.2.

<u>ALGORITHM</u> <u>5.3</u>: Evaluating a B-Spline Expansion and Its Derivatives

```
Input:
        k               the order of the spline
        x               the evaluation point
        t[n+k]          the knot vector
        a[1:n,0:k-1]    the differenced basis coefficients

Temporaries:
        dp[k-1]     dm[k-1]      v[k]

Output:
        i               the integer interv( x )
        d[0:k-1]        the spline and its k-1 derivatives
                        evaluated at x

Algorithm:

1        i = interv( x )

COMMENT  Compute the B-Splines

2        v[1] := 1
3        FOR r:=1 UNTIL k-1 DO
4         | dp[r] := t[i+r] - x
5         | dm[r] := x - t[i+1-r]
6         | vp := 0
```



```
7        | FOR  s:= 1 UNTIL r  DO
8        |  [ vm := v[s]/(dp[s] + dm[r+1-s])
9        |  | v[s] := vp + dp[s]*vm
10       |  L vp := dm[r+1-s]*vm
11       | v[r+1]:= vp

COMMENT  | Compute the Derivatives

12       | sum := 0
13       | FOR  j:= 0 UNTIL r  DO
14       |  [ sum := sum + a[i-r+j,k-1-r]*v[j+1]
15       L d[k-1-r] := sum
16       d[k-1] := a[i,k-1]
```

Operation counts for the algorithms described in this section are summarized in Table 5.1. The storage requirements listed apply only to each processing step. The total storage required for a computation is the sum of the expressions listed for each of the steps.

TABLE 5.1

Storage and Operation Counts for
Evaluating B-Splines and B-Spline Expansions

| Algorithm | Description | Operations | Storage |
|---|---|---|---|
| | Storing $\underline{a}$<br>Storing $\underline{t}$ | | $n$<br>$n+k$ |
| 5.1 | Nonvanishing B-splines of orders $1,\ldots,k$ | $\frac{3}{2}k(k-1)$ | $k(k+2)$ |
| 5.2 | Nonvanishing B-splines of order $k$ | $\frac{3}{2}k(k-1)$ | $3k$ |
| | Preprocessing: $\underline{a}^{(\ell)}$<br>Evaluating $\ell^{th}$ derivative of a spline | $2n\ell$<br>$\frac{3}{2}(k-\ell)(k-\ell-1)+k-\ell$ | $n$ |
| 5.3 | Preprocessing: $\underline{a}^{(\ell)}$ $1\leq \ell \leq k-1$<br>Evaluating a spline and all $k-1$ derivatives at a point | $2n(k-1)$<br>$2k(k-1)+k$ | $n(k-1)$<br>$4k$ |



III.6 The Piecewise Polynomial Representation

Splines can also be represented as piecewise polynomials. Evaluating a piecewise polynomial ($k-1$ multiplications) is much less costly than evaluating a B-spline expansion ($\frac{1}{2}(3k^2-k)$ multiplications). Since converting a B-spline expansion to a piecewise polynomial is cheap ($\sim 2nk^2$ multiplications, see §7), if a spline will be evaluated at more than $2n$ points, then the spline should be evaluated from its piecewise polynomial representation, even if it was originally represented as a B-spline expansion.

The coefficient piecewise polynomial representation for a spline $s(t) \in S(k,\underline{t})$ consists of the knot vector $\underline{u}$ and the polynomial coefficient matrix

$$(6.1)\quad C \equiv [\, c_{i,j} \,], \qquad c_{i,j} \equiv \frac{D^j s(u_i^+)}{j!}, \qquad 0\leq i \leq m, \quad 0\leq j \leq k-1.$$

The values of the spline and its derivatives are given by

$$(6.2)\quad D^\ell s(t) = \sum_{j=\ell}^{k-1} \frac{j!}{(j-\ell)!} c_{i,j} (t-u_i)^{j-\ell},$$

$$u_i \leq t \leq u_{i+1}, \quad 0\leq i \leq m, \quad 0\leq \ell \leq k-1.$$

Evaluating a spline using Horner's rule requires $k-1$ multiplications (because $\frac{j!}{(j-\ell)!} = 1$ for $\ell = 0$); and evaluating the $\ell^{th}$ derivative, $1\leq \ell \leq k-1$, requires $2(k-\ell-1)$ multiplications.

**ALGORITHM 6.1:** Coefficient Piecewise Polynomial Evaluation

Input:

| | |
|---|---|
| k | the order of the spline |
| m | the number of interior knots |
| $\ell$ | the derivative to be evaluated |
| u[0:m+1] | the knots |
| x | the evaluation point |
| C[0:m,0:k-1] | the polynomial coefficient array |
| $\frac{j!}{(j-\ell)!}$ | a precomputed table, $\ell \le j \le k-1,\ 1 \le \ell \le k-1$. |

Output:

value

**Algorithm:**

```
compute i, 0≤ i ≤m  such that  u[i]≤ t ≤u[i+1]
dx = x-u[i]
IF  ℓ:=0 THEN
   value := C[i,k-1]
   FOR j:=k-2 STEP -1 UNTIL 0 DO
      value := value*dx + C[i,j]
ELSE
   value := C[i,k-1]*(k-1)!/(k-1-ℓ)!
   FOR j:=k-2 STEP -1 UNTIL ℓ DO
      value := value*dx + C[i,j]*j!/(j-ℓ)!
```

Alternatively, as in the "B-spline Code" [B4], the piecewise polynomial could be represented by the knot vector $\underline{u}$ and the polynomial derivative matrix

$$(6.3)\quad C^D \equiv [\, c^D_{i,j} \,], \qquad c^D_{i,j} \equiv D^j s(u_i^+), \qquad 0 \le i \le m, \quad 0 \le j \le k-1.$$



In this representation, the values of the spline and its derivatives are given by the Taylor's series expansion

$$(6.4)\quad D^\ell s(t) = \sum_{j=\ell}^{k-1} \frac{c^D_{i,j}}{(j-\ell)!} (t-u_i)^{j-\ell},$$

$$u_i \le t \le u_{i+1}, \qquad 0 \le i \le m, \qquad 0 \le \ell \le k-1.$$

Evaluating the $\ell^{th}$ derivative, $0 \le \ell \le k-1$, requires $2(k-1-\ell)$ multiplications. Although the algorithm for evaluating derivatives from this representation is somewhat shorter, evaluating the spline itself requires twice as many multiplications.

**ALGORITHM 6.2:** Derivative Piecewise Polynomial Evaluation

Input:

| | |
|---|---|
| k | the order of the spline |
| m | the number of interior knots |
| $\ell$ | the derivative to be evaluated |
| u[0:m+1] | the knots |
| x | the evaluation point |
| $C^D$[0:m,0:k-1] | the derivative array |

Output:

value

**Algorithm:**

```
compute i, 0≤ i ≤m  such that  u[i]≤ t ≤u[i+1]
dx = x-u[i]
value := C^D[i,k-1]
FOR j:=k-2 STEP -1 UNTIL ℓ DO
   value := value * dx/(j-ℓ+1) + C^D[i,j]
```

If the evaluation points are uniformly spaced and numerous, then a very efficient incremental evaluation scheme can be employed. Suppose

that the spline is to be evaluated at the points

$$x_i = x_0 + i\delta, \quad 0 \le i \le M,$$

all of which lie in the same interval of $\underline{u}$. In that interval, the spline is a k-1 degree polynomial which can be computed by integrating the first order system

$$\begin{aligned} \delta\, Ds_0(t) &= s_1(t) \\ \delta\, Ds_1(t) &= s_2(t) \\ &\cdots \\ \delta\, Ds_{k-2}(t) &= s_{k-1}(t) = \delta^{k-1} Ds_{k-1}(x_0) \end{aligned}$$

with the initial conditions

$$\begin{aligned} s_0(t) &= \delta^0 \quad s_0(x_0) \\ s_1(t) &= \delta^1 \quad Ds_1(x_0) \\ &\cdots \\ s_{k-2}(t) &= \delta^{k-2} Ds_{k-1}(x_0) . \end{aligned}$$

An approximate integration of this system at the points $x_i$, $1 \le i \le M$, can be obtained from the Euler iteration

$$\begin{aligned} y_i^{(0)} &= y_{i-1}^{(0)} + y_{i-1}^{(1)} \\ y_i^{(1)} &= y_{i-1}^{(1)} + y_{i-1}^{(2)} \\ &\cdots \\ y_i^{(k-2)} &= y_{i-1}^{(k-2)} + y_{i-1}^{(k-1)}, \quad 1 \le i \le M. \end{aligned}$$

Since the Euler iteration is exact for linear polynomials [D1,§8.2], the iterates $y_i^{(k-1)}$ and $y_i^{(k-2)}$ are exact. However, for $k \ge 3$, the iterates $y_i^{(\ell)}$, $0 \le \ell < k-3$, may give only approximate values. This incremental evaluation scheme requires k-1 additions per point.



While these iterates are generally not exact, they are polynomials in i, i.e., (see [K1,§1.2.6,(9),(40)])

$$(6.5) \quad y_i^{(\ell)} = \sum_{j=\ell}^{k-1} \binom{i}{j-\ell} y_0^{(j)} ,$$

$$= \sum_{\xi=0}^{k-1-\ell} i^{\xi} \sum_{j=\ell+\xi}^{k-1} \frac{(-1)^{j-\ell-\xi}}{(j-\ell)!} y_0^{(j)} \left[{j-\ell \atop \xi}\right], \quad 1 \le i \le M, \ 0 \le \ell \le k-1,$$

where the $\left[{j-\ell \atop \xi}\right]$ are the Stirling numbers of the first kind (see [K1,§1.2.6,(40)]). Consequently, an exact evaluation scheme can be obtained by choosing the initial conditions so that $y_i^{(0)}$ is the desired k-1 degree polynomial $s(x_i)$.

Clearly, (see [K1,§1.2.6,(41)])

$$s(x_i) = \sum_{\xi=0}^{k-1} i^{\xi} \frac{\delta^{\xi} D^{\xi} s(x_0)}{\xi!}$$

$$= \sum_{j=0}^{k-1} \binom{i}{j} \sum_{\xi=j}^{k-1} \frac{j!}{\xi!} \left\{{\xi \atop j}\right\} \delta^{\xi} D^{\xi} s(x_0) ,$$

where the $\left\{{\xi \atop j}\right\}$ are the Stirling numbers of the second kind (see [K1,§1.2.6,(41)]). Consequently, from (6.5) with $\ell = 0$, if the initial conditions for the Euler iteration are changed to

$$y_0^{(j)} = \sum_{\xi=j}^{k-1} \frac{j!}{\xi!} \left\{{\xi \atop j}\right\} \delta^{\xi} D^{\xi} s(x_0) , \quad 0 \le j \le k-1.$$

then $y_i^{(0)} = s(x_i)$, $0 \le i \le M$, and this scheme will be exact except for roundoff error. Many of the iterates for the higher derivatives will still not be exact, i.e.,

$$y_i^{(\ell)} \ne D^{\ell} s(x_i), \quad 1 \le i \le M, \quad 1 \le \ell \le k-2,$$

for all but certain fortuitous choices of initial conditions.

<u>ALGORITHM 6.3</u>: Incremental Evaluation of a Spline

Input:

| | |
|---|---|
| $k$ | the order of the spline |
| $M$ | the number of evaluation points |
| $x_0$ | the starting point |
| $\delta$ | the spacing between evaluation points |
| $D^{\ell}s(x_0)$ | the derivatives of the spline at $x_0$ |
| $\left\{ {\xi \atop j} \right\}$ | a precomputed table of Stirling numbers |

Temporary
F[0:k-1]

Output:
$s(x_0+i\delta)$, $0 \le i \le M$, through the routine PUT

<u>Algorithm</u>:

```
COMMENT  Initialization

FOR j:=0 UNTIL k-1 DO
  [ sum := 0
  [ FOR ξ:=j UNTIL k-1 DO
  [   [ sum := sum + (j!/ξ!) {ξ j} δ^ξ D^ξ s(x_0)
  [ F_j := sum

COMMENT  Iteration

PUT( F_0 )
FOR i:=1 UNTIL M
  [ FOR ℓ:=k-2 STEP -1 UNTIL 0 DO
  [   [ F_ℓ := F_ℓ + F_ℓ+1
  [ PUT( F_0 )
```



Operation counts for the algorithms described in this section are summarized in Table 6.1.

<u>TABLE 6.1</u>

Storage and Operation Counts
for Evaluating Piecewise Polynomials

(Operations counted are multiplications or divisions unless otherwise noted.)

| Algorithm | Description | Operations | Storage |
|---|---|---|---|
| | Storing u[0:m+1]<br>Storing C[0:m,0:k-1] | | m+2<br>k(m+1) |
| 6.1<br>(coeff.) | Evaluating spline<br>Evaluating $D^{\ell}$ | k-1<br>2(k-ℓ) | |
| 6.2<br>(deriv.) | Evaluating $D^{\ell}$ | 2(k-ℓ) | |
| 6.3 | Uniform spacing $\delta$ | k-1 additions | k |

## III.7 Converting to a Piecewise Polynomial

In this section, two different conversion algorithms are presented. The first, and most general, algorithm is a specialized version of Algorithm 5.3. The second algorithm is a table look-up algorithm for the special case of uniform knot spacing. Both algorithms require $O(nk^2)$ operations and $O(k^2)$ locations (not including storage for the piecewise polynomial).

The piecewise polynomial corresponding to a B-spline expansion can be obtained by evaluating the spline's derivatives at the knots using Algorithm 5.3 and computing the polynomial coefficients from (6.1). The resulting algorithm is composed of three parts: computing the nonvanishing B-spline values in the triangle (5.2), computing the differenced basis coefficients using (5.4), and computing the polynomial coefficients using (6.1).

Because many of the B-splines in (5.2) vanish at the knots, operations involving these B-splines can be avoided. In particular, (e.g., Figure 7.1),

$$N_{i,r}(t_i) = 0, \qquad 2 \le r \le k,$$

$$N_{i,1}(t_i) = N_{i-1,2}(t_i) = 1, \quad i \text{ such that } t_i < t_{i+1},\ k \le i \le n.$$

The $\frac{1}{2}(k-1)(k-2)-1$ remaining values in the triangle (5.2) can be computed using the the following specialized version of Algorithm 5.1. This algorithm requires $\frac{3}{2}(k-1)(k-2)$ multiplications, $3(k-1)$ fewer than Algorithm 5.1 (one-half fewer for $k = 4$).



TABLE 7.1

The B-Splines Evaluated at $t_i$ for Uniform Knot Spacing

| | $N_{i-5,k}$ | $N_{i-4,k}$ | $N_{i-3,k}$ | $N_{i-2,k}$ | $N_{i-1,k}$ | $N_{i,k}$ |
|---|---|---|---|---|---|---|
| k | | | | | | |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 2 | 0 | 0 | 0 | 0 | 1 | 0 |
| 3 | 0 | 0 | 0 | 1/2 | 1/2 | 0 |
| 4 | 0 | 0 | 1/6 | 2/3 | 1/6 | 0 |
| 5 | 0 | 1/24 | 11/24 | 11/24 | 1/24 | 0 |
| 6 | 1/120 | 13/60 | 11/20 | 13/60 | 1/120 | 0 |

ALGORITHM 7.1: Evaluating the B-splines at a Knot

```
Input:
        k               the order of the spline
        t[n+k]          the knot vector
        i               a knot index satisfying t_i < t_i+1, k<= i <=n

Temporaries:
        dp[k-1], dm[k-1]

Output:
        N[k,k]          the values
                          N[j,r] = N_i-r+j,r(x),  1<= j <=r,  1<= r <=k.

Algorithm:

        N[1,1] := 1
        N[1,2] := 1
        dp[1] := t[i+1] - t[i]
        FOR r:=2 UNTIL k-1 DO
         | dp[r] := t[i+r] - t[i]
         | dm[r] := t[i] - t[i+1-r]
         | N[1,r+1] := 0
         | FOR s:= 1 UNTIL r-1 DO
         |  | vm := N[s,r]/(dp[s] + dm[r+1-s])
         |  | N[s,r+1] := N[s,r+1] + dp[s]*vm
         |  | N[s+1,r+1] := dm[r+1-s]*vm
```

To save $n(k-1)$ multiplications, we compute the basis coefficient difference array $A \equiv [\, a_{j,\ell} \,]n\times(k-1)$, where

$$(7.1)\qquad \begin{aligned} a_{j,0} &\equiv a_j, \qquad 1\le j \le n. \\ a_{j,\ell} &\equiv (k-\ell)\,\frac{a_{j,\ell-1} - a_{j-1,\ell-1}}{t_{j+k-\ell} - t_j}, \\ &\qquad \ell+1\le j \le n,\quad t_{j+k-\ell}-t_j>0,\quad 1\le \ell \le k-1, \end{aligned}$$

instead of computing $\underline{a}^{(\ell)}$. Consequently, the piecewise polynomial coefficients are given by (see (5.3), (5.4) and (6.1))

$$(7.2)\qquad c_{i,\ell} \equiv \binom{k-1}{\ell} \sum_{j=i-k+\ell+1}^{i} a_{j,\ell}\, N_{j,k-\ell}(t_i), \qquad 0\le i \le m,\quad 0\le \ell \le k-1.$$

The cost of computing the difference array A varies from as few as $\frac{1}{2}(m+1)k(k-1)$ multiplications for $z = k$ to fewer than $n(k-1)$ multiplications for $z = 1$.

To save storage (at the cost of some indexing overhead), we store only the rows of A required to compute the piecewise polynomial in each interval, i.e., while computing the polynomial representing the spline in $[t_i,t_{i+1})$, only rows $i-k+1, \ldots, i$ of A are stored. Since multiplications by vanishing B-spline values are avoided, computation of the piecewise polynomial coefficients using (7.2) requires only $\frac{1}{2}(m+1)(k^2+k-4)$ multiplications. The complete algorithm requires fewer than $(m+1)(2k^2-4k+1) + n(k-1)$ multiplications



<u>ALGORITHM 7.2</u>: Conversion to a Piecewise Polynomial

Input:

| | |
|---|---|
| $k \ge 2$ | the order of the spline |
| t[n+k] | the B-spline knot vector |
| a[n] | the basis coefficient vector |
| $\binom{k-1}{\ell}$ | a precomputed binomial coefficient table |

Output:

| | |
|---|---|
| C[0:m,0:k-1] | the piecewise polynomial array |
| u[0:m+1] | the piecewise polynomial knot vector |

Temporaries

| | |
|---|---|
| A[0:k-1,0:k-1] | rows i-k+1 through i of the difference array A |
| N[k,k] | the B-spline basis functions (5.2) |

<u>Algorithm</u>:

```
        i := k;   low := 1;   ic := 0
        WHILE i ≤ n DO
COMMENT  Locate a nondegenerate interval of t
          IF t[i] < t[i+1] THEN
            u[ic] := t[i]
            Use Algorithm 7.1 to compute the nonvanishing B-splines.
            Store the results in N[k,k].
COMMENT    Compute a segment of A
            FOR j:=low UNTIL i DO
              pj :=  mod(j-1,k)
              A[pj,0] := a[j]
              FOR ℓ:=1 UNTIL (k-1)-(i-j) DO
                A[pj,ℓ] := (A[pj,ℓ-1] - A[pj-1,ℓ-1]) / (t[j+k-ℓ] - t[j])
COMMENT    Compute piecewise polynomial coefficients
            FOR ℓ:=0 UNTIL k-3 DO
              sum := 0
              FOR j:=1 UNTIL k-1-ℓ DO
                pj := mod( i-(k-ℓ)+j-1,k)
                sum := sum  + A[pj,ℓ]*N[j,k-ℓ]
              C[ic,ℓ] := sum*(k-1 choose ℓ)
            C[ic,k-2] := (k-1 choose 1)*A[i-1,k-2]
            C[ic,k-1] := A[i,k-1]
            low := i+1;  ic:=ic+1
          i := i+1;
        u[m+1] := t[n+1]
```

If the knots are uniformly spaced, i.e., if

$$t_i = ih, \quad 1 \le i \le n+k,$$

then the conversion algorithm can be simplified greatly. The k-order B-splines can be written as the scaled translates

$$N_{i,k}(t) \equiv B_k\left(\frac{t-t_i}{h}\right)$$

of a canonical k-order B-spline [P4,p.90]

$$B_k(x) \equiv \begin{cases} \frac{1}{(k-1)!} \sum_{j=0}^{k} \binom{k}{j} (-1)^j (x-j)_+^{k-1}, & 0 \le x \le k, \\ 0, & \text{Otherwise} \end{cases}$$

Moreover, if $C_k^B[0:k-1,0:k-1]$ is an array containing the piecewise polynomial for this canonical B-spline (see Table 7.2), then

$$N_{i,k}(t) \equiv \begin{cases} \sum_{\ell=0}^{k-1} C_k^B[j,\ell] \left(\frac{t-t_{i+j}}{h}\right)^{\ell}, & t \in [t_{i+j}, t_{i+j+1}], \quad 0 \le j \le k-1, \\ 0, & \text{Otherwise} \end{cases}$$

The conversion algorithm is a simple matrix-vector multipication requiring $(m+1)k^2$ operations, half as many as Algorithm 7.2.



TABLE 7.2

The Canonical B-spline Piecewise Polynomials $C_k^B[0:k-1,0:k-1]$

| k | j \ ℓ: | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | | | | | |
| 2 | 0 | 0 | 1 | | | | |
| | 1 | 1 | -1 | | | | |
| 3 | 0 | 0 | 0 | 1/2 | | | |
| | 1 | 1/2 | 1 | -1 | | | |
| | 2 | 1/2 | -1 | 1/2 | | | |
| 4 | 0 | 0 | 0 | 0 | 1/6 | | |
| | 1 | 1/6 | 1/2 | 1/2 | -1/2 | | |
| | 2 | 2/3 | 0 | -1 | 1/2 | | |
| | 3 | 1/6 | -1/2 | 1/2 | -1/6 | | |
| 5 | 0 | 0 | 0 | 0 | 0 | 1/24 | |
| | 1 | 1/24 | 1/6 | 1/4 | 1/6 | -1/6 | |
| | 2 | 11/24 | 1/2 | -1/4 | -1/2 | 1/4 | |
| | 3 | 11/24 | -1/2 | -1/4 | 1/2 | -1/6 | |
| | 4 | 1/24 | -1/6 | 1/4 | -1/6 | 1/24 | |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 1/120 |
| | 1 | 1/120 | 1/24 | 1/12 | 1/12 | 1/24 | -1/24 |
| | 2 | 13/60 | 5/12 | 1/6 | -1/6 | -1/6 | 1/12 |
| | 3 | 11/20 | 0 | -1/2 | 0 | 1/4 | -1/12 |
| | 4 | 13/60 | -5/12 | 1/6 | 1/6 | -1/6 | 1/24 |
| | 5 | 1/120 | -1/24 | 1/12 | -1/12 | 1/24 | -1/120 |

ALGORITHM 7.3: Conversion to Piecewise Polynomial, Translates

Input:

```
k                  the order of the spline
n                  the number of basis coefficients
a[n]               the basis coefficient array
h                  the knot spacing
```

$C_k^B[0:k-1,0:k-1]$ the B-spline piecewise polynomials

Temporary:

$C_h[1:k,0:k-1]$ the array $C_k^B$ scaled by powers of h

Output:

C[0:m,0:k-1] the piecewise polynomial for the spline

Algorithm:

```
COMMENT    Scale the Canonical B-Spline Array by Powers of h

     hh := 1
     FOR ℓ:=0 UNTIL k-1 DO
       FOR i:=1 UNTIL k DO
           C_h[i,ℓ] := C_k^B[k-i,ℓ]*hh
       hh := hh/h

COMMENT    Compute Matrix-Vector Product

     FOR ℓ:=0 UNTIL k-1 DO
       FOR i:=0  UNTIL n-k DO
           sum := 0
           FOR j:=1 UNTIL k DO
               sum := sum + C^N[j,ℓ]*a[i+j]
           C[i,ℓ] := sum
```



Operation counts for the algorithms described in this section are summarized in Table 7.3.

TABLE 7.3

Storage and Operation Counts
for Conversion to Piecewise Polynomials

| Algorithm | Description | Operations | Storage |
|---|---|---|---|
| | Storing u[0:m+1]<br>Storing C[0:m,0:k-1] | | m+2<br>k(m+1) |
| 7.1 | Evaluating B-splines at a knot | $\frac{3}{2}(k-1)(k-2)$ | $\sim k^2$ |
| 7.2 | Conversion to piecewise polynomial | $n(k-1)$<br>$+(m+1)(2k^2-4k+1)$ | $\sim 3k^2$ |
| 7.3 | Conversion to piecewise polynomial (uniform knot spacing) | $(m+2)k^2$ | $\sim 2k^2$ |

## III.8 Forming the Normal Equations (Faster)

In step 2b of Algorithm 2.1 (the basic least-squares spline algorithm), the B-splines $N_{i-k+1,k}(x_\ell)$, ... $N_{i,k}(x_\ell)$, i = interv($x_\ell$), $1 \leq \ell \leq N$, are evaluated. Since these B-splines are also splines, any of the spline evaluation schemes in §5 or §6 can be used. In this section,

we describe and analyze several fast schemes for evaluating these B-splines.

The simplest and most general of these schemes is Algorithm 5.2. Only 3k storage locations (in addition to $\underline{t}$) and $3Nk(k-1)/2$ operations are needed to evaluate the non-vanishing B-splines required for Algorithm 2.1. Forming the normal equations requires a total of $2Nk^2$ operations (see Table 8.2).

To speed up the B-spline computation we could convert all of the B-splines into piecewise polynomials. Neglecting conversion cost, evaluating the B-splines needed for Algorithm 2.1 requires $Nk(k-1)$ operations. Forming the normal equations requires a total of $N(3k^2+k)/2$ operations, approximately one-quarter fewer than using Algorithm 5.2 (see Table 8.2).

The B-splines can be converted to piecewise polynomials using a version of Algorithm 7.3 specially adapted for the B-splines. Because the arrays involved are sparse, this algorithm requires far fewer operations than the straightforward application of Algorithm 7.3. Large temporary arrays are avoided by storing only the elements of the arrays required to compute the B-spline piecewise polynomials in each interval.

For each B-spline, at most $k(k+1)/2$ of the entries in the difference array $A[1:n,0:k-1]$ are nonzero (e.g., Figure 8.1) and computing the nonvanishing elements of these arrays for all n B-splines requires at most $\frac{1}{2}n(k^2+k-2)$ operations. Moreover, because many of the B-splines vanish at the knots (e.g., Figure 8.2), the sums in (6.3) involve only a small fraction of the elements in A (the elements labeled



"+" in Figure 8.1). In general, the sum in (7.2) requires (see Figure 8.1, Figure 8.2, and [K1,§1.2.6,Fig.6])

$$\begin{aligned} &1 + 2 + \quad + k-3 + k-2 + k-1 + \\ &1 + 2 + \ldots + k-3 + k-2 + \\ &\quad \ldots \qquad\qquad\qquad - 2(k-1) \;=\; \binom{k+1}{3} - 2(k-1) \\ &1 + 2 + 3 + 4 + \\ &1 + 2 + 3 + \\ &1 + 2 + \\ &1 \\ &\qquad\qquad\qquad\qquad = \frac{k(k-1)(k+1)}{6} - 2(k-1) \end{aligned}$$

multiplications and the leading term of (7.2) requires another $(k-1)$ multiplications.

FIGURE 8.1

The Difference Arrays $A[1:n,0:k-1]$ for the k = 5 B-splines

| | $N_{i-4,5}$ | | | | | $N_{i-3,5}$ | | | | | $N_{i-2,5}$ | | | | | $N_{i-1,5}$ | | | | | $N_{i,5}$ | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $\ell$ = | 0 | 1 | 2 | 3 | 4 | 0 | 1 | 2 | 3 | 4 | 0 | 1 | 2 | 3 | 4 | 0 | 1 | 2 | 3 | 4 | 0 | 1 | 2 | 3 | 4 | |
| Row | | | | | | | | | | | | | | | | | | | | | | | | | | Row |
| i-5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | i-5 |
| i-4 | 1 | x | x | x | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | i-4 |
| i-3 | 0 | + | x | x | x | 1 | + | x | x | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | i-3 |
| i-2 | 0 | 0 | + | x | x | 0 | + | + | x | x | 1 | + | + | x | x | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | i-2 |
| i-1 | 0 | 0 | 0 | + | x | 0 | 0 | + | + | x | 0 | + | + | + | x | 1 | + | + | + | x | 0 | 0 | 0 | 0 | 0 | i-1 |
| i | 0 | 0 | 0 | 0 | + | 0 | 0 | 0 | x | + | 0 | 0 | x | x | + | 0 | x | x | x | + | 1 | x | x | x | + | i |
| i+1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | 0 | 0 | 0 | x | x | 0 | 0 | x | x | x | 0 | x | x | x | x | i+1 |
| i+2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | 0 | 0 | 0 | x | x | 0 | 0 | x | x | x | i+2 |
| i+3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | 0 | 0 | 0 | x | x | i+3 |
| i+4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | x | i+4 |
| i+5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | i+5 |

x -- a non-zero value
\+ -- needed to compute the polynomial representing the B-spline in $[t_i,t_{i+1}]$

**FIGURE 8.2**

The B-Splines Evaluated at the Knot $t_i$

$$
\begin{array}{c|ccccc|c}
 & & & N_{j,r} & & & \\
 & r=5 & 4 & 3 & 2 & 1 & \\
j & & & & & & j \\
i-5 & 0 & 0 & 0 & 0 & 0 & i-5 \\
i-4 & x & 0 & 0 & 0 & 0 & i-4 \\
i-3 & x & x & 0 & 0 & 0 & i-3 \\
i-2 & x & x & x & 0 & 0 & i-2 \\
i-1 & x & x & x & 1 & 0 & i-1 \\
i & 0 & 0 & 0 & 0 & 1 & i \\
i+1 & 0 & 0 & 0 & 0 & 0 & i+1
\end{array}
$$

x -- a non-zero value

Using Algorithm 7.1, computing the k non-vanishing B-splines at each of the knots requires $\frac{3}{2}(m+1)(k-1)(k-2)$ operations. Thus, computing the piecewise polynomials for all of the B-splines requires a total of $\frac{1}{2}n(k^2+k-2) + \frac{m+1}{6}(k^3+9k^2-34k+24)$ multiplications. (See Table 8.1 and Figure 8.3 for a comparison of Algorithm 7.2 and Algorithm 5.2.)

**TABLE 8.1**

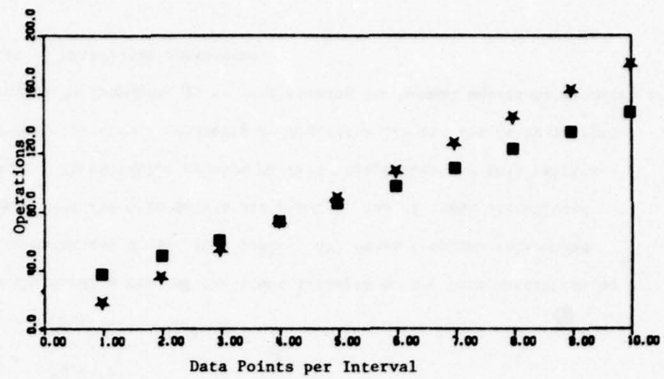
Breakeven Points for B-Spline Piecewise Polynomial Conversion
Large n and $\underline{z} = 1$

| k | Data Points/Interval |
|---|---|
| 2 | 2.0 |
| 3 | 3.3 |
| 4 | 4.2 |
| 5 | 4.8 |
| 6 | 5.3 |
| 7 | 5.8 |
| 8 | 6.3 |
| 9 | 6.7 |
| 10 | 7.1 |



**FIGURE 8.3**

Operation Counts to Evaluate k = 4 B-splines for large n and z = 1 with (★) and without (■) converting to piecewise polynomials




**ALGORITHM 8.1**: B-Spline Conversion to a Piecewise Polynomial

Input:

| | |
|---|---|
| $k \geq 2$ | the order of the spline |
| $t[n+k]$ | the knot vector |
| $\binom{k-1}{\ell}$ | a precomputed binomial coefficient table |

Temporaries

| | |
|---|---|
| A[0:k-1,-1:k-1,0:k-1] | the difference array |
| N[k,k] | the B-spline basis functions |
| C[0:k-1] | the polynomial representing a B-spline in one interval |

Output:

The piecewise polynomials for the B-splines in each interval through the PUT function

**Algorithm:**

```
COMMENT   Initialize the Basis Coefficient Array

1         FOR i:=0 UNTIL k-1 DO
2           FOR j:=-1 UNTIL k-1 DO
3             A[0,j,i] := 0
4           A[i,0,0] := 1

5         i := k;   low := 1;  ic := 1

COMMENT   Loop through knots

6         WHILE i<n DO

COMMENT     Locate a nondegenerate interval of t

7           IF t[i] < t[i+1] THEN
8               PUT( t[i] );

COMMENT     Compute the B-splines not vanishing at t[i]

9           Use Algorithm 7.1 to compute the nonvanishing B-splines
            Store the results in N[k,k].

COMMENT     Basis Function Loop

10            FOR p:=i-k+1 UNTIL i-1 DO
11              pp = mod( p-1,k )
12              FOR j:=max(low,p) UNTIL i DO
13                pj :=j-p
14                FOR ℓ:=max(1,pj) UNTIL  (k-1)-(i-j) DO
15                  A[pp,pj,ℓ] :=  (A[pp,pj,ℓ-1]-A[pp,pj-1,ℓ-1]) / (t[j+k-ℓ]-t[j])
COMMENT     Compute piecewise polynomial coefficients

16              C[0]   := N[p-(i-k),k]
17              FOR ℓ:=1 UNTIL k-3 DO
18                sum := 0
19                FOR j:=max(p,i-k+ℓ) UNTIL i-1 DO
20                  sum := sum  + A[pp,j-p,ℓ]*N[j-(i-k+ℓ),k-ℓ]
21                C[ℓ] := sum*(k-1 over ℓ)
22              C[k-2] := (k-1 over 1)*A[pp,i-1-p,k-2]
23              C[k-1] :=        A[pp,i-p,k-1]
24              PUT( C[0], ...,  C[k-1] )

25            pp = mod(i-1,k)
26            PUT( 0, ..., 0,  A[pp,0,k-1] )
27            low := i;  ic:=ic+1

28          i := i+1;
```



In the special case of knots with uniform spacing h, we can precompute tables of the B-spline piecewise polynomials (e.g., Table 7.1), and there is no B-spline conversion cost. Moreover, if the data are uniformly weighted and uniformly spaced with respect to the knots, with M points in each interval, i.e., if

$$\underline{x} = \begin{pmatrix} x_1, & x_2, & \ldots, & x_M \\ h+x_1, & h+x_2, & \ldots, & h+x_M \\ 2h+x_1, & 2h+x_2, & \ldots, & 2h+x_M, \\ \vdots & \vdots & \vdots & \end{pmatrix}$$

and

$$x_\ell \in [t_k, t_{k+1}], \quad 1 \le \ell \le M ,$$

then the necessary B-spline values

$$N_{1,k}(x_\ell),$$
$$N_{2,k}(x_\ell),$$
$$\ldots$$
$$N_{k,k}(x_\ell), \quad 1 \le \ell \le M,$$

and the first k rows of the lower triangle of the Gram matrix can be precomputed and stored in a table. All other B-spline values and elements of the Gram matrix are equal to one of these precomputed values. These tables require $Mk(2k-1)$ operations and $Mk+k^2$ storage. The only operations remaining in Algorithm 2.1 are the $Nk$ operations involved in computing $\underline{b}$, so that forming the normal equations requires a total of $Mk(k+1)+Nk$ operations.

**ALGORITHM 8.2:** Computing $\underline{b}$, Uniform Data and Knot Spacing

Input:

| | |
|---|---|
| k | the order of the spline |
| n | the number of basis coefficients |
| M | the number of data points per interval |
| y[M,n-k+1] | the data ordinates |
| N[M,k] | a precomputed table $N[\ell,j] = N_{j,k}(x_\ell)$, $1 \le \ell \le M$, $1 \le j \le k$ |

Output:

| | |
|---|---|
| b[n] | the right hand side |

Algorithm:

```
FOR i:=1 UNTIL n DO
  b[i] := 0
FOR i:=1 UNTIL n-k+1 DO
  FOR j:=1 UNTIL k DO
    FOR ℓ:=1 UNTIL M DO
      b[i+j-1] := b[i+j-1] + N[ℓ,j]*y[ℓ,i]
```



Operation counts for some of the algorithms described in this section are summarized in Table 8.2.

TABLE 8.2

Storage and Operation Counts for Forming the Normal Equations

| Algorithm | Description | Operations | Storage |
|---|---|---|---|
| | Storing G | | nk |
| | Storing $\underline{b}$ | | n |
| 5.2 | Computing B-splines | $\frac{3}{2}$ Nk(k-1) | |
| 2.1 | Computing G terms | $\frac{1}{2}$ Nk(k+1) | |
| 2.1 | Computing $\underline{b}$ terms | Nk | |
| | TOTAL | 2 N $k^2$ | |
| 8.1 | Conversion | $\sim 2nk^2 + \frac{n^3}{6}$ | $k^3+k(k+3)$ |
| 6.1 | Computing B-splines | Nk(k-1) | |
| 2.1 | Computing G terms | $\frac{1}{2}$ Nk(k+1) | |
| 2.1 | Computing $\underline{b}$ terms | Nk | |
| | TOTAL | $\frac{1}{2}$ Nk(3k+1) | |
| 6.1 | Computing B-splines for uniform knot spacing | k(k-1)N | $k^2$ |
| 2.1 | Computing G terms | $\frac{1}{2}$ Nk(k+1) | |
| 2.1 | Computing $\underline{b}$ terms | Nk | |
| | TOTAL | $\frac{3}{2}$ Nk(k+1) | |
| | B-spline tables | $\frac{3}{2}$ Mk(k-1) | M k |
| | G tables | $\frac{1}{2}$ Nk(k+1) | $k^2$ |
| 8.2 | Computing $\underline{b}$ terms | Nk | |
| | TOTAL | Nk + Mk(2k-1) | |

## Chapter IV

## Local Dependence and Linear Systems

### IV.1 Introduction

The B-spline Gram matrices are symmetric, banded, positive definite, and well-conditioned, independent of order (see §II.3 and §II.5). These matrices satisfy a set of local dependence results which may not apply to more general classes of matrices. In later chapters, we will employ these matrix local dependence results to develop simple local dependence bounds for least-squares splines (§V.3,§V.4), local error bounds for least-squares splines (§V.5), and efficient, limited-storage algorithms for computing least-squares splines (Chapter VI). In this chapter, we establish the foundations by developing a unified theory of local dependence for symmetric, positive definite, block tridiagonal matrices, independent of spline approximation theory.

The basis for these local dependence results is an "exponential damping" bound on the blocks of the inverse matrix of the form

$$\| (A^{-1})_{i,j} \|_2 \le K \gamma^{|i-j|},$$

for some $\gamma < 1$ depending on the condition number of A. Such bounds have been derived for many special classes of matrices, including diagonally dominant, tridiagonal matrices [K3], spline interpolation matrices





[A2,K3,L2,S2], and B-spline Gram matrices [P2,D7,D8,D9,B5,D3,D4]. More recently, Hager and Strang [H1] have developed exponential damping bounds for diagonally dominant band matrices, and Demko [D4] has derived similar bounds for more general band matrices. In §3, we employ an argument similar to that of Hager and Strang [H1] to derive exponential damping bounds for symmetric, positive definite, block tridiagonal matrices.

The matrix local dependence results of §4-§7 follow from this exponential damping bound. In §4, we develop bounds on the change in elements of the solution of a linear system $A\,\underline{x} = \underline{b}$ due to local changes in the matrix A, local changes in $\underline{b}$, and nonlocal changes in $\underline{b}$. For example, the change in element $a_i$ due to a change in element $b_j$ is bounded by an expression of the form $K\,\gamma^{|i-j|}$. Early results of this type were derived by Ahlberg, Nilson, and Walsh [A2] and Powell [P2] for certain special linear systems involved in spline interpolation and least-squares spline approximation.

In §5, we develop error bounds for local solutions to linear systems. If the linear system $A\,\underline{x} = \underline{b}$ is partitioned as

$$\begin{bmatrix} A' & \cdot \\ \cdot & \cdot \end{bmatrix} \begin{bmatrix} \underline{x}' \\ \cdot \end{bmatrix} = \begin{bmatrix} \underline{b}' \\ \cdot \end{bmatrix},$$

then the local solution $\underline{x}^L$ is defined to be the solution to the local linear system

$$A'\,\underline{x}^L = \underline{b}'\ .$$

In general we would not expect the local solution $\underline{x}^L$ to be related to the solution subvector $\underline{x}'$. However, if A is a symmetric, positive

definite, and block tridiagonal matrix, then the $i^{th}$ element of the error $e_i \equiv x_i^L - x_i'$ is bounded by an expression of the form $K \, {}^{r-i}$, where r is the order of A′. Consequently, except for the last few elements, the local solution $\underline{x}^L$ is a good approximation to $\underline{x}'$, and we can compute accurate estimates for any part of the solution to a large linear system by solving the appropriate local linear system. Local computation algorithms based on this observation can be employed in real-time or limited-storage applications where there is insufficient main memory to store the entire matrix (see Chapter VI and [E2,L2,E4]).

In §6 and §7, we develop error bounds for local inverses and local Cholesky factorizations of matrices. If the matrix A is partitioned as

$$A = \begin{bmatrix} \cdot & \cdot \\ \cdot & A'' \end{bmatrix},$$

then $(A'')^{-1}$ is a local inverse of A and $M M^T = A''$ is a local Cholesky factorization of A. In both cases, the difference between an element of the local inverse or local Cholesky factor and the corresponding element of the inverse or Cholesky factorization of A is bounded by an exponential in the distance from the upper left corner of A″. An interesting consequence of these results is that the inverse and Cholesky factorization of a matrix composed of rows of identical blocks can be computed to high accuracy using a small fixed number of operations, independent of the order of the matrix. Results of this type have been developed for the special case of point tridiagonal matrices by Malcolm and Palmer [M2].



IV.2 <u>Definitions</u>

Throughout this chapter we will employ standard linear algebra notation [e.g., F1,S8,V1]. The notation developed for least-squares spline approximation in Chapter II and Chapter III will have no special significance.

Let A denote the symmetric, positive definite, block tridiagonal matrix

$$(2.1)\quad A \equiv \begin{bmatrix} D_1 & C_1 & & & \bigcirc \\ C_1^T & D_2 & C_2 & & \\ & \ddots & \ddots & \ddots & \\ & & C_{n-2}^T & D_{n-1} & C_{n-1} \\ \bigcirc & & & C_{n-1}^T & D_n \end{bmatrix}.$$

Clearly, A = D + C, where D is a matrix containing only the diagonal blocks of A, and C is a matrix containing only the off-diagonal blocks. In the following lemma we summarize some of the important properties of the eigenvalues of A and of the associated matrix D − C.

<u>LEMMA 2.1</u>

If

$$(2.2)\quad S \equiv \begin{bmatrix} I_1 & & & \bigcirc \\ & -I_2 & & \\ & & \ddots & \\ \bigcirc & & & (-1)^n I_n \end{bmatrix},$$

is a block diagonal matrix with the same partitioning as A, then

(2.3) $S = S^{-1}$

and A is similar to D − C, i.e.,

(2.4) $S A S = D - C$ .

Moreover, if $\lambda$ is an eigenvalue of $A = D + C$ with corresponding eigenvector $\underline{x}_\lambda$, then $\lambda$ is an eigenvalue of $D - C$ with corresponding eigenvector $S \underline{x}_\lambda$.

<u>Proof:</u> Equations 2.3 and 2.4 are easy to verify. Moreover, if $\lambda$ is an eigenvalue of A, then

$$A \underline{x}_\lambda = \lambda \underline{x}_\lambda ,$$

and

$$(S^{-1} A S) S^{-1} \underline{x}_\lambda = \lambda S^{-1} \underline{x}_\lambda .$$

Thus, since $S = S^{-1}$ and $S A S = D - C$,

$$(D - C)(S \underline{x}_\lambda) = \lambda (S \underline{x}_\lambda),$$

i.e., $\lambda$ is also an eigenvalue of $D - C$ and $S \underline{x}_\lambda$ is the corresponding eigenvector.

Q.E.D.

For any matrix B partitioned as in (2.1), let $B_{i,j}$ denote the i,j submatrix (or block) of B and $\mathrm{ord}(B_{i,i})$ denote the number of rows (and columns) in $B_{i,i}$. If a vector $\underline{x}$ is partitioned into subvectors consistent with (2.1), then let the vector $\underline{x}_i$ denote the $i^{th}$ such subvector.



We will employ the usual matrix and vector norms as well as the mixed $2,\infty$ norm

$$\| \underline{x} \|_{2,\infty} \equiv \max_{1 \le j \le n} \| \underline{x}_j \|_2 .$$

The spectral radius $\rho(B)$ is defined as

$$\rho(B) \equiv \max( |\lambda_{min}(B)|, |\lambda_{max}(B)| ).$$

Alternatively, if B is symmetric, then

$$\rho(B) = \| B \|_2 = \lambda_{max}(B).$$

## IV.3 <u>An Exponential Damping Bound</u>

In this section we derive a bound on the $\ell_2$ norms of the submatrices of $A^{-1}$. The argument is based on the Neumann series for $A^{-1}$ [S8,p.191] and the special properties of symmetric, positive definite, block tridiagonal matrices.

Since the matrix A is positive definite, the matrices D and $D^{-1}$ are also positive definite and there are block diagonal "square root" matrices P such that $P P^T = D^{-1}$, e.g., the Cholesky decomposition of $D^{-1}$ [F1,§3; S8,p.140] . For all such "square root" matrices, $A^{-1}$ can be written as

$$A^{-1} = ( D + C )^{-1} = ( P^{-T} P^{-1} + C )^{-1} = P ( I + P^T C P )^{-1} P^T ,$$

or the Neumann series [S8,p.191]

(3.1) $$A^{-1} = P \left( \sum_{k=0}^{\infty} (-P^T C P)^k \right) P^T ,$$

provided this series converges. However, the Neumann series (3.1) converges if and only if [S8,p.191]

$$(3.2)\quad \gamma \equiv \rho(\,D^{-1}C\,) = \rho(\,P^TC\,P\,) < 1, \qquad P\,P^T = D^{-1}.$$

*In the following lemma we show that* $\gamma$ *is strictly less than unity* for symmetric, positive definite, block tridiagonal matrices. (Note that this result can also be viewed as a proof of convergence for the block Jacobi iteration [cf. Y1,p.214].)

<u>LEMMA 3.1</u>

If A is a symmetric, positive definite, block tridiagonal matrix, then

$$(3.3)\quad \gamma = \frac{\kappa(P^TA\,P) - 1}{\kappa(P^TA\,P) + 1} < 1\,, \qquad P\,P^T = D^{-1}.$$

<u>Proof:</u> Clearly,

$$P^TA\,P = P^T\,(D + C)\,P = I + P^TC\,P.$$

Moreover, applying Lemma 2.1 to $P^TC\,P$,

$$\begin{aligned}\lambda_{min}(\,P^TC\,P\,) &= \lambda_{max}(-P^TC\,P\,)\\ &= -\lambda_{max}(\,P^TC\,P\,)\\ &= -\rho(\,P^TC\,P\,),\end{aligned}$$

so that, from [S8,p.266] and (3.2), the minimum and maximum eigenvalues of $P^TA\,P$ are

$$\lambda_{min}(\,P^TA\,P\,) = 1 - \rho(\,P^TC\,P\,) \equiv 1 - \gamma$$
$$\lambda_{max}(\,P^TA\,P\,) = 1 + \rho(\,P^TC\,P\,) \equiv 1 + \gamma.$$



Since $P^TA\,P$ is symmetric and positive-definite [S8,p.189,p.308],

$$\kappa(\,P^TA\,P\,) = \frac{\lambda_{max}(\,P^TA\,P\,)}{\lambda_{min}(\,P^TA\,P\,)} = \frac{1+\gamma}{1-\gamma}$$

and the result follows.

Q.E.D.

To bound $\gamma$ using (3.3), we must bound the condition number of the the matrix $P^TA\,P$. In general, this task may be difficult. However, in the following result we show that $\gamma$ can be bounded by an expression involving the condition number of any block-diagonal scaling of A.

<u>LEMMA 3.2</u>

If W is a nonsingular matrix which commutes with the matrix S defined in (2.2), then

$$(3.4)\quad \gamma \le \frac{\kappa(W^TA\,W) - 1}{\kappa(W^TA\,W) + 1},$$

with equality if $W\,W^T = D^{-1}$.

<u>Proof:</u> For notational convenience, define

$$\bar{C} \equiv W^T C\, W,$$

$$\bar{D} \equiv W^T D\, W,$$

$$\bar{A} \equiv W^T A\, W = W^T\,(D + C)\,W = \bar{D} + \bar{C}$$

Noting that W commutes with S, and applying Lemma 2.1, we obtain

$$\begin{aligned} S^{-1}\bar{A}\,S &= S^T W^T A\, W\, S \\ &= W^T S^T A\, S\, W \\ &= W^T (D - C)\, W \\ &= \bar{D} - \bar{C}\ , \end{aligned}$$

so that $\bar{D} - \bar{C}$ is similar to $\bar{A}$.

The Rayleigh quotient of $\bar{A}$ is bounded above and below by the largest and smallest eigenvalues of $\bar{A}$ [S8,p.312]. Moreover, because $\bar{D} - \bar{C}$ is similar to $\bar{A} = \bar{D} + \bar{C}$, the Rayleigh quotient of $\bar{D} - \bar{C}$ is also bounded above and below by the largest and smallest eigenvalues of $\bar{A}$. Thus,

$$\lambda_{min}(\bar{A})\ \underline{x}^T\underline{x} \ \le\ \underline{x}^T(\bar{D} \pm \bar{C})\ \underline{x} \ \le\ \lambda_{max}(\bar{A})\ \underline{x}^T\underline{x},$$

After multiplying the left-hand inequality by $\lambda_{max}$,

$$\lambda_{max}\lambda_{min}\ \underline{x}^T\underline{x} \ \le\ \lambda_{max}\ \underline{x}^T\bar{D}\ \underline{x} - \lambda_{max}\ \underline{x}^T\bar{C}\ \underline{x}\ ;$$

multiplying the right-hand inequality by $-\lambda_{min}$,

$$-\lambda_{max}\lambda_{min}\ \underline{x}^T\underline{x} \ \le\ -\lambda_{min}\ \underline{x}^T\bar{D}\ \underline{x} - \lambda_{min}\ \underline{x}^T\bar{C}\ \underline{x}\ ;$$

and summing the two resulting inequalities, we obtain

$$(3.5)\quad (\lambda_{max} + \lambda_{min})\ \underline{x}^T\bar{C}\ \underline{x} \ \le\ (\lambda_{max} - \lambda_{min})\ \underline{x}^T\bar{D}\ \underline{x}\ .$$

Similarly, after multiplying the right-hand inequality by $\lambda_{min}$, multiplying the left-hand inequality by $-\lambda_{max}$, and summing the two resulting inequalities, we obtain

$$(3.6)\quad -(\lambda_{max} - \lambda_{min})\ \underline{x}^T\bar{D}\ \underline{x} \ \le\ (\lambda_{max} + \lambda_{min})\ \underline{x}^T\bar{C}\ \underline{x}\ .$$



Making the substitution $\underline{x} = Q\,\underline{z}$ with $Q\,Q^T = \bar{D}^{-1}$, we can combine (3.5) and (3.6) to obtain

$$-(\lambda_{max}-\lambda_{min})\ \underline{z}^T\underline{z} \le (\lambda_{max}+\lambda_{min})\ \underline{z}^T Q^T \bar{C}\ Q\ \underline{z} \ \le (\lambda_{max}-\lambda_{min})\ \underline{z}^T\underline{z}.$$

and [S8,p.314]

$$\rho(\,Q^T\bar{C}\,Q\,) \le \frac{\lambda_{max}(\bar{A}) - \lambda_{min}(\bar{A})}{\lambda_{max}(\bar{A}) + \lambda_{min}(\bar{A})} = \frac{\kappa(\bar{A}) - 1}{\kappa(\bar{A}) + 1}\ .$$

Since $(W\,Q)^T(W\,Q) = D^{-1}$,

$$\gamma \equiv \rho(\,(W\,Q)^T\,C\,(W\,Q)\,) = \rho(\,Q^T\bar{C}\,Q\,)$$

and the result follows.

Q.E.D.

Incidentally, since

$$g(x) \equiv \frac{x - 1}{x + 1}, \qquad x \ge 1,$$

is a monotonically increasing function of x, the inequality

$$\kappa(\,P^T A\,P\,) \ \le\ \kappa(\,W^T A\,W\,), \qquad P\,P^T = D^{-1}, \quad W\,S = S\,W,$$

follows from (3.3) and (3.4). Thus, $P^T A\,P$ is an optimal block $\ell_2$-scaling of A and we have derived the following block generalization of an optimal $\ell_2$ scaling result of Forsythe and Strauss [F3].

THEOREM 3.3

If A is a symmetric, positive definite, and block tridiagonal matrix and W is a nonsingular matrix W which commutes with S, then

$$\kappa( P^T A\, P ) \le \kappa( W^T A\, W ) ,$$

with equality if $W W^T = D^{-1}$.

Because the matrix $P^T C\, P$ is block tridiagonal, the terms of the Neumann series (3.1) have a special nonzero structure (see Lemma 6.2 for more details). In particular, a simple induction argument implies that the i,j block of the $k^{th}$ term $(-P^T C\ P)^k$ is zero for $|i-j| > k$. Thus, we can write (3.1) as

$$(3.7)\quad (A^{-1})_{i,j} \equiv P_i \left( \sum_{k=|i-j|}^{\infty} ([-P^T C\ P]^k)_{i,j} \right) P_j^T ,$$

where the lower limit of the sum in (3.1) has been changed to $|i-j|$ from 0. From the triangle inequality

$$(3.8)\quad \| (A^{-1})_{i,j} \|_2 \le \|P_i\|_2 \left( \sum_{k=|i-j|}^{\infty} \| ([-P^T C\ P]^k)_{i,j} \|_2 \right) \|P_j^T\|_2 .$$

To obtain the desired "exponential damping" bound on $\| (A^{-1})_{i,j} \|_2$, we employ the following lemma to bound the $\ell_2$-norms of the submatrices $\| ([-P^T C\ P]^k)_{i,j} \|_2$ in terms of the norms of the matrices $\| P^T C\ P \|_2^k$.



LEMMA 3.4

For any symmetric matrix B,

$$(3.9)\quad \|B_{i,j}\|_2 \le \| B \|_2, \qquad 1 \le i,j \le n.$$

If B is also positive definite, then

$$(3.10)\quad \|B_{i,j}\|_2 \le \tfrac{1}{2} \| B \|_2, \qquad 1 \le i \ne j \le n.$$

Proof: For any symmetric matrix B [S8,p.312],

$$(3.11)\quad \lambda_{min}(B)\, \underline{x}^T\underline{x} \le \underline{x}^T B\, \underline{x} \le \lambda_{max}(B)\, \underline{x}^T\underline{x}.$$

By choosing the vector $\underline{x}$ appropriately, we can obtain bounds on the eigenvalues of the submatrices of B. We use these bounds to obtain bounds on the spectral norms of the submatrices.

First, we bound the norms of the diagonal matrices $B_{i,i}$, $1 \le i \le n$. For any vector $\underline{y}$ of length $\mathrm{ord}(B_{i,i})$, let the vector $\underline{x}$ in (3.11) be

$$\underline{x} = [\ \underline{x}_\ell\ ], \quad \text{where } \underline{x}_\ell = \begin{cases} \underline{y} & \text{if } \ell = i \\ \underline{0} & \text{otherwise} \end{cases}, \quad 1 \le \ell \le n .$$

Then

$$\lambda_{min}\, \underline{y}^T\underline{y} \le \underline{y}^T B_{i,i}\, \underline{y} \le \lambda_{max}\, \underline{y}^T\underline{y}$$

and [S8,p.308]

$$\|B_{i,i}\|_2 = \rho(B_{i,i}) \le \max(|\lambda_{min}(B)|, |\lambda_{max}(B)|) = \rho(B) = \|B\|_2,$$

which is (3.9) for $i = j$.

Second, we bound the norms of the off-diagonal submatrices $B_{i,j}$, $1 \le i,j \le n$. For any vector $\underline{y}$ of length $\mathrm{ord}(B_{i,i})$ and any vector $\underline{z}$ of length $\mathrm{ord}(B_{j,j})$, let the vector $\underline{x}$ in (3.11) be

$$\underline{x} = [\,\underline{x}_\ell\,], \quad \text{where } \underline{x}_\ell = \begin{cases} \underline{y} & \text{if } \ell = i \\ \underline{z} & \text{if } \ell = j \\ \underline{0} & \text{otherwise} \end{cases}, \quad 1 \le \ell \le n .$$

Then

$$(3.12)\quad \lambda_{min}(\underline{y}^T\underline{y}+\underline{z}^T\underline{z}) \le \underline{y}^T B_{ii}\,\underline{y} + 2\underline{y}^T B_{ij}\,\underline{z} + \underline{z}^T B_{jj}\,\underline{z} \le \lambda_{max}(\underline{y}^T\underline{y}+\underline{z}^T\underline{z}) .$$

Similarly, for the same vectors $\underline{y}$ and $\underline{z}$, let the vector $\underline{x}$ in (3.11) be

$$\underline{x} = [\,\underline{x}_\ell\,], \quad \text{where } \underline{x}_\ell = \begin{cases} -\underline{y} & \text{if } \ell = i \\ \underline{z} & \text{if } \ell = j \\ \underline{0} & \text{otherwise} \end{cases}, \quad 1 \le \ell \le n,$$

so that

$$(3.13)\quad \lambda_{min}(\underline{y}^T\underline{y}+\underline{z}^T\underline{z}) \le \underline{y}^T B_{ii}\,\underline{y} - 2\underline{y}^T B_{ij}\,\underline{z} + \underline{z}^T B_{ii}\,\underline{z} \le \lambda_{max}(\underline{y}^T\underline{y}+\underline{z}^T\underline{z}) .$$

Subtracting (3.13) from (3.12), we find that

$$(3.14)\quad -(\lambda_{max}-\lambda_{min})(\underline{y}^T\underline{y}+\underline{z}^T\underline{z}) \le 4\underline{y}^T B_{ij}\,\underline{z} \le (\lambda_{max}-\lambda_{min})(\underline{y}^T\underline{y}+\underline{z}^T\underline{z}) .$$



From (3.14) and [S8,p.180],

$$\|B_{i,j}\|_2 = \max_{\|y\|_2 = \|z\|_2 = 1} \underline{y}^T B_{i,j}\,\underline{z} = \tfrac{1}{2}(\lambda_{max} - \lambda_{min}) \le \tfrac{1}{2}(|\lambda_{max}| + |\lambda_{min}|) \le \rho(B) = \|B\|_2,$$

which proves (3.9) for $i \ne j$. For positive definite matrices, $\lambda_{min} > 0$ and

$$\|B_{i,j}\|_2 = \tfrac{1}{2}(\lambda_{max} - \lambda_{min}) \le \tfrac{1}{2}\lambda_{max} = \tfrac{1}{2}\|B\|_2 ,$$

which proves (3.10).

Q.E.D.

THEOREM 3.5

If A is a symmetric, positive definite, block tridiagonal matrix, then

$$\|(A^{-1})_{i,j}\|_2 \le \|D^{-1}\|_2 (1-\gamma)^{-1} \gamma^{|i-j|}, \quad 1 \le i,j \le n,$$

where $\gamma$ is defined in (3.2).

Proof: From (3.8) and Lemma 3.4,

$$\|(A^{-1})_{i,j}\|_2 \le \|P\|_2^2 \sum_{k=|i-j|}^{\infty} \|P^T C P\|_2^k .$$

Moreover, from the definition of the 2-norm [S8,p.180],

(3.15) $\| F \|_2^2 = \| P P^T \|_2 = \| D^{-1} \|_2$

and because C is symmetric,

(3.16) $\| P^T C P \|_2 = \rho( P^T C P ) \equiv \gamma$ .

Thus,

$$\| (A^{-1})_{i,j} \|_2 \leq \| D^{-1} \|_2 \sum_{k=|i-j|}^{\infty} \gamma^k ,$$

and the result follows from the simple relations

$$\sum_{k=|i-j|}^{\infty} \gamma^k = \gamma^{|i-j|} \sum_{\ell=0}^{\infty} \gamma^{\ell} = (1-\gamma)^{-1} \gamma^{|i-j|} .$$

Q.E.D.

## IV.4 Local Dependence of Solutions to Linear Systems

Theorem 3.5 leads to bounds on changes in elements of the solution of a linear system due to local changes in the coefficient matrix or the right-hand side. The first result (simple local dependence) is a bound on the norm of the difference between the elements of the solution to the linear system

(4.1) $A \underline{x} = \underline{b}$

and the elements of the solution to the locally perturbed linear system

(4.2) $A \underline{x}^{\delta} = \underline{b} + \underline{\delta}$,



where, for some fixed j, the local perturbation $\underline{\delta}$ is given by

$$\underline{\delta} \equiv [ \underline{\delta}_i ], \quad \underline{\delta}_i \equiv \begin{cases} \underline{\varepsilon} \text{ for } i=j \\ \underline{0} \text{ otherwise} \end{cases}, \quad 1 \leq i \leq n.$$

COROLLARY 4.1

If $\underline{x}^{\delta}$ is the solution to the perturbed linear system (4.2), then

$$\| \underline{x}_i^{\delta} - \underline{x}_i \|_2 \leq \| D^{-1} \|_2 (1-\gamma)^{-1} \gamma^{|i-j|} \| \underline{\varepsilon} \|_2, \quad 1 \leq i \leq n.$$

Proof: The result follows from Theorem 3.5 and the identity

$$\underline{x}_i^{\delta} - \underline{x}_i = (A^{-1})_{i,j} \underline{\delta}_j, \quad 1 \leq i \leq n.$$

Q.E.D.

The second result (matrix local dependence) is a bound on the difference between the elements of the solution to (4.1) and the elements of the solution to the locally perturbed linear system

(4.3) $(A + \Delta) \underline{x}^{\Delta} = \underline{b}$,

where for fixed j and k, the perturbation matrix $\Delta$ is given by

$$\Delta \equiv [\Delta_{\ell,m}]_{n \times n}, \quad \Delta_{\ell,m} \equiv \begin{cases} E \text{ if } \ell=j \text{ and } m=k \\ 0 \text{ otherwise} \end{cases}, \quad 1 \leq \ell,m \leq n.$$

Note that the perturbation matrix $\Delta$ need not be symmetric.

COROLLARY 4.2

If $\underline{x}^{\Delta}$ is the solution to the perturbed linear system (4.3), then

$$\|\underline{x}_i^{\Delta} - \underline{x}_i\|_2 \le \|D^{-1}\|_2 (1-\gamma)^{-1} \gamma^{|i-j|} \|E\|_2 \|\underline{x}_k^{\Delta}\|_2, \quad 1 \le i \le n.$$

Proof: The result follows from the identity

$$A\underline{x}^{\Delta} = \underline{b} - \Delta\underline{x}^{\Delta}$$

and Corollary 4.1 with

$$\underline{\varepsilon} = E\underline{x}_k^{\Delta}.$$

Q.E.D

The final local dependence result is a bound on the change in the solution due to nonlocal perturbations in the vector $\underline{b}$. The effect of such perturbations on the elements of the solution is also bounded by an exponential in the distance from the perturbation. The perturbed linear system is of the form

(4.4) $A\underline{x}^{\beta} = \underline{b} + \underline{\beta}$,

where for some fixed integer $r$, $0 \le r \le n$, the first $r$ elements of the perturbation $\underline{\beta}$ are 0, i.e.,

$$\underline{\beta}_i = \underline{0}, \quad 1 \le i \le r.$$



THEOREM 4.3

If $\underline{x}^{\beta}$ is the solution to the perturbed linear system (4.4), then

(4.5) $\|\underline{x}_i^{\beta} - \underline{x}_i\|_2 < 2\|D^{-1}\|_2(1-\gamma)^{-2}\|\underline{\beta}\|_{2,\infty}, \quad 1 \le i \le n,$

and

(4.6) $\|\underline{x}_i^{\beta} - \underline{x}_i\|_2 < \|D^{-1}\|_2(1-\gamma)^{-2}\|\underline{\beta}\|_{2,\infty}\gamma^{1+r-i}, \quad 1 \le i \le r.$

Proof: Clearly,

$$\underline{x}_i^{\beta} - \underline{x}_i = \sum_{j=r+1}^{n} (A^{-1})_{i,j}\underline{\beta}_j.$$

From the triangle inequality,

$$\|\underline{x}_i^{\beta} - \underline{x}_i\|_2 \le \sum_{j=r+1}^{n} \|(A^{-1})_{i,j}\|_2 \|\underline{\beta}_j\|_2, \quad 1 \le i \le n.$$

Applying Theorem 3.5, we obtain

$$\|\underline{x}_i^{\beta} - \underline{x}_i\|_2 \le \|D^{-1}\|_2(1-\gamma)^{-1}\left(\sum_{j=r+1}^{n} \gamma^{|i-j|}\right)\|\underline{\beta}\|_{2,\infty}, \quad 1 \le i \le n.$$

If $i \le r$, then

(4.7) $\sum_{j=r+1}^{n} \gamma^{|i-j|} < \gamma^{1+r-i} \sum_{\ell=0}^{\infty} \gamma^{\ell} = \gamma^{1+r-i}(1-\gamma)^{-1},$

and we obtain (4.6). Otherwise,

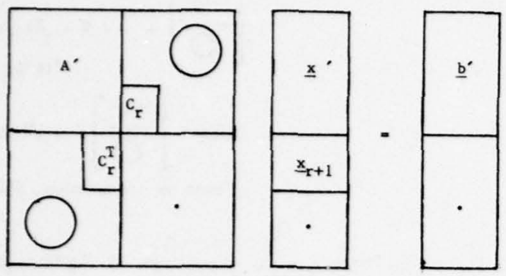
(4.8) $\sum_{j=r+1}^{n} \gamma^{|i-j|} < \sum_{\ell=-\infty}^{+\infty} \gamma^{|\ell|} = 2(1-\gamma)^{-1} - 1 < 2(1-\gamma)^{-1},$

and we obtain (4.5).

Q.E.D.

## IV.5 Local Solutions to Linear Systems

If the linear system $A\,\underline{x} = \underline{b}$ is partitioned as

(5.1) 


where

$$(5.2)\quad A' = \begin{bmatrix} D_1 & C_1 & & & \bigcirc \\ C_1^T & D_2 & C_2 & & \\ & \ddots & \ddots & \ddots & \\ & & C_{r-2}^T & D_{r-1} & C_{r-1} \\ \bigcirc & & & C_{r-1}^T & D_r \end{bmatrix},$$

(5.3) $\underline{x}' = (\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_r)$ ,

(5.4) $\underline{b}' = (\underline{b}_1, \underline{b}_2, \ldots, \underline{b}_r)$ ,

then the local solution $\underline{x}^L$ is the solution to the local linear system

(5.5) $A'\,\underline{x}^L = \underline{b}'$ .

In this section, we obtain bounds on the elements of the error vector $\underline{e}^L \equiv \underline{x}^L - \underline{x}'$.



Before proceding to the major results, we will derive some useful bounds on the parameters of a diagonal block-submatrix B of A in terms of the corresponding quantities for the full matrix A. These bounds enable us, for any diagonal block-submatrix B in the subsequent analysis, to bound $\gamma_B$ by $\gamma \equiv \gamma_A$, $(1-\gamma_B)^{-1}$ by $(1-\gamma)^{-1}$, and $\| D_B^{-1} \|_2$ by $\| D^{-1} \|_2 \equiv \| D_A^{-1} \|_2$.

LEMMA 5.1

If B is a diagonal block-submatrix of A, then

(5.6) $\| D_B^{-1} \|_2 \leq \| D^{-1} \|_2 \equiv \| D_A^{-1} \|_2$

(5.7) $\gamma_B \leq \gamma \equiv \gamma_A$ .

Proof: Since B is a diagonal block submatrix of A, from [S8,p.266,p.308],

$$\begin{aligned} \| D_B^{-1} \|_2 &= \max_{D_i \in D_B} \| D_i^{-1} \|_2 \\ &\leq \max_{D_i \in D_A} \| D_i^{-1} \|_2 \\ &= \| D_A^{-1} \|_2 \end{aligned}$$

and (5.6) follows. Moreover, if $P_B$ is the submatrix of P corresponding to B, then $P_B^T B\, P_B$ is a diagonal block-submatrix of $P^T B\, P$ and from Lemma 3.4,

$$\| P_B^T B\, P_B \|_2 \leq \| P^T A\, P \|_2 .$$

Similarly,

$$\| (P_B^T B\, P_B)^{-1} \|_2 \leq \| (P^T A\, P)^{-1} \|_2$$

so that

$$\kappa(P_B^T B\ P_B) = \frac{\|\ P_B^T B\ P_B\ \|_2}{\|\ (P_B^T B\ P_B)^{-1}\ \|_2} \leq \frac{\|\ P^T A\ P\ \|_2}{\|\ (P^T A\ P)^{-1}\ \|_2} = \kappa(P^T A\ P).$$

Finally, from (3.3), $\gamma$ is a monotonic increasing function of the condition number of $\kappa(\ P^T A\ P\ )$ and we obtain (5.7).

Q.E.D.

The following result provides a bound on the difference between the elements of the local solution $\underline{x}^L$ and the corresponding elements of the solution $\underline{x}$.

LEMMA 5.2

If $\underline{x}^L$ is a local solution to the linear system $A\ \underline{x} = \underline{b}$, then

$$\|\underline{x}_i^L - \underline{x}_i\|_2 \leq \kappa(D)\ (1-\gamma)^{-1}\ \gamma^{r+1-i}\ \|\underline{x}_{r+1}\|_2, \qquad 1 \leq i \leq r.$$

Proof: From (5.1), we have

$$A'\ \underline{x}' + \begin{bmatrix} O \\ C_r \underline{x}_{r+1} \end{bmatrix} = \underline{b}'$$

and from (5.5),

$$A'\ (\underline{x}^L - \underline{x}') = - \begin{bmatrix} O \\ C_r \underline{x}_{r+1} \end{bmatrix}.$$



Thus,

$$\underline{x}_i^L - \underline{x}_i' = ((A')^{-1})_{i,r}\ C_r\ \underline{x}_{r+1}$$

and

$$\|\underline{x}_i^L - \underline{x}_i\|_2 \leq \|((A')^{-1})_{i,r}\|_2\ \|C_r\|_2\ \|\underline{x}_{r+1}\|_2, \quad 1 \leq i \leq r.$$

From Lemma 3.4, (3.15), and (3.16),

$$\begin{aligned} (5.8) \quad \|C_r\|_2 &\leq \|\ C\ \|_2 \\ &= \|\ P^{-T}\ P^T\ C\ P\ P^{-1}\ \|_2 \\ &\leq \|\ P^{-T}\|_2\ \ \|\ P^{-1}\|_2\ \ \|\ P^T\ C\ P\ \|_2 \\ &= \gamma\ \|\ D\ \|_2, \end{aligned}$$

so that

$$\|\underline{x}_i^L - \underline{x}_i\|_2 \leq \gamma\ \|\ D\ \|_2\ \|((A')^{-1})_{i,r}\|_2\ \|\underline{x}_{r+1}\|_2,$$

and the result follows from Theorem 3.5.

Q.E.D

The bound of Theorem 5.2 is a posteriori, i.e., it is written in terms of the solution vector. In the following theorem, we obtain an a priori bound by applying Theorem 3.5 to the result of Lemma 5.2

<u>THEOREM 5.3</u>

If $\underline{x}^L$ is a local solution to the linear system $A\,\underline{x} = \underline{b}$, then

$$\| \underline{x}_i^L - \underline{x}_i \|_2 < 2\,\kappa(D)\ \|D^{-1}\|_2\,(1-\gamma)^{-3}\,\gamma^{r+1-i}\,\|\underline{b}\|_{2,\infty}, \quad 1 \le i \le r.$$

<u>Proof</u>: From Lemma 5.2 and the triangle inequality,

$$\| \underline{x}_i^L - \underline{x}_i \|_2 \le \kappa(D)\,(1-\gamma)^{-1}\,\gamma^{r+1-i}\ \| (A^{-1}\underline{b})_{r+1} \|_2,$$

$$\le \kappa(D)\,(1-\gamma)^{-1}\,\gamma^{r+1-i} \sum_{j=1}^{n} \|(A^{-1})_{r+1,j}\|\ \|\underline{b}_j\|_2 .$$

Applying Theorem 3.5, we obtain

$$\| \underline{x}_i^L - \underline{x}_i \|_2 \le \kappa(D)\,(1-\gamma)^{-2}\ \|D^{-1}\|_2\,\gamma^{r+1-i}\ \|\underline{b}\|_{2,\infty} \sum_{j=1}^{n} \gamma^{|r+1-j|} .$$

The result follows from (4.8).

Q.E.D

## IV.6 <u>Local Inverses of Matrices</u>

If the matrix A is partitioned as in (5.1), then $(A')^{-1}$ is a (lower) local inverse of A. The error in submatrices of $(A')^{-1}$ could be bounded by applying the local solution results of §5 to the linear system $A\,X = I$. However, in this section we derive somewhat stronger bounds by returning to the basic argument of Equation 3.8 and Theorem 3.5. We also show that, if the matrix A is composed of identical block rows, then the block rows of $A^{-1}$ are nearly identical. Consequently, for any fixed precision, there are only a fixed number of significantly different blocks in $A^{-1}$ and the inverse of A can be



computed in a fixed number of arithmetic operations, independent of the order of A.

As an approximation to $(A^{-1})_{i,j}$, we take the i,j submatrix of the $\ell^{th}$ partial sum of the expansion (3.7), i.e.,

$$(6.1)\quad (A^{-1})_{i,j} \approx M(\ell,i,j) \equiv P_i \left( \sum_{k=|i-j|}^{\ell} ([-P^T C\ P]^k)_{i,j} \right) P_j^T .$$

In §3, we observed that $M(\ell,i,j)$ converges to $(A^{-1})_{i,j}$ for large $\ell$. The actual rate of convergence is bounded in the following result.

<u>LEMMA 6.1</u>

If A is a symmetric, positive definite, block tridiagonal matrix, then

$$\|M(\ell,i,j)-(A^{-1})_{i,j}\|_2 \le \|D^{-1}\|_2\,(1-\gamma)^{-1}\,\gamma^{\ell+1}, \quad \ell \ge |i-j|,\ 1 \le i,j \le n,$$

where $\gamma$ is defined in (3.2).

<u>Proof:</u> From (6.1) and the triangle inequality,

$$\| M(\ell,i,j) - (A^{-1})_{i,j} \|_2$$

$$\le \| P_i \|_2 \left\{ \sum_{k=\ell+1}^{\infty} \|([P^T C\ P]^k)_{i,j}\|_2 \right\} \| P_j^T \|_2 .$$

The result follows as in the proof of Theorem 3.5.

Q.E.D.

Let $M'(\ell,i,j)$ be the series approximation to a block of the local inverse $([A']^{-1})_{i,j}$, $1 \le i,j \le r$. In the following result, we give nontrivial choices for $\ell$ such that $M'(\ell,i,j) = M(\ell,i,j)$.

LEMMA 6.2

If A is a symmetric, positive definite, block tridiagonal matrix, then

$$M'(\ell,i,j) = M(\ell,i,j), \quad 0 \le \ell \le 2r - (i+j), \quad 1 \le i,j \le r.$$

Proof: Consider the $k^{th}$ terms of the sums (6.1) for A and A′. Because C and C′ are both block tridiagonal matrices,

$$([-P^T C\ P]^k)_{i,j} = ([-(P')^T C'\ P']^k)_{i,j} = 0, \quad i+j+k \text{ even.}$$

Moreover, if i+j+k is odd, then by a simple induction argument,

$$([-P^T C\ P]^k)_{i,j} = ([-(P')^T C'\ P']^k)_{i,j}, \quad i+j+k \le 2r.$$

Consequently, if $i+j+\ell \le 2r$, then all of the corresponding terms of the sums (6.1) for $M'(\ell,i,j)$ and $M(\ell,i,j)$ are equal and

$$M'(\ell,i,j) = M(\ell,i,j) .$$

Q.E.D.

An error bound for the local inverse follows directly from Lemma 6.1 and Lemma 6.2.

THEOREM 6.3

If A′ is a local inverse of A, then

$$\|([A']^{-1})_{i,j} - (A^{-1})_{i,j}\|_2 \le 2\,\|D^{-1}\|_2\,(1-\gamma)^{-1}\,\gamma^{1+2r-(i+j)}, \quad 1 \le i,j \le r.$$

Proof: From (6.1) and Lemma 6.2,

$$([A']^{-1})_{i,j} = M'(2r-(i+j),i,j) = M(2r-(i+j),i,j) = (A^{-1})_{i,j}.$$

Using Lemma 6.1 to bound the approximation errors, we obtain

$$\|M(2r-(i+j),i,j) - (A^{-1})_{i,j}\|_2 \le \|D^{-1}\|_2 (1-\gamma)^{-1}\,\gamma^{1+2r-(i+j)}$$



and

$$\|M'(2r-(i+j),i,j) - ([A']^{-1})_{i,j}\|_2 \le \|(D')^{-1}\|_2 (1-\gamma')^{-1}(\gamma')^{1+2r-(i+j)}.$$

From Lemma 5.1, the second inequality can be written as

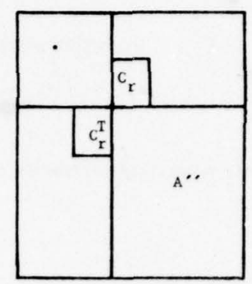
$$\|M(2r-(i+j),i,j) - ([A']^{-1})_{i,j}\|_2 \le \|D^{-1}\|_2\,(1-\gamma)^{-1}\,\gamma^{1+2r-(i+j)}$$

and the result follows from the triangle inequality.

Q.E.D.

This error bound can be extended to other block partitionings of A. If the matrix A is partitioned as

(6.2) A =




,

then $(A'')^{-1}$ is an (upper) local inverse of A. An error bound for the upper local inverse can be derived from Theorem 6.2 by transposing A and A′′ about the minor diagonal.

COROLLARY 6.4

If $A''$ is an upper local inverse of A, then

$$\|([A'']^{-1})_{i-r,j-r}-(A^{-1})_{i,j}\|_2 \le 2\,\|D^{-1}\|_2(1-\gamma)^{-1}\,\gamma^{i+j-2r-1}, \quad r+1\le i,j\le n.$$

Frequently matrices are composed of identical block rows, i.e.,

$$(6.3)\qquad A \equiv \begin{bmatrix} X & Y & & & \bigcirc \\ Y^T & X & Y & & \\ & \ddots & \ddots & \ddots & \\ & & Y^T & X & Y \\ \bigcirc & & & Y^T & X \end{bmatrix}.$$

The inverses of such matrices are composed of nearly identical block rows, i.e., each submatrix $(A^{-1})_{i,j}$, $1\le i,j\le n$, *approximates the* corresponding submatrix on the minor diagonal $(A^{-1})_{k,m}$, $k+m = n+1$, $|k-m| = |i-j|$.

COROLLARY 6.5

If A is composed of identical block rows, *then*

$$\|(A^{-1})_{i,j}-(A^{-1})_{k,m}\|_2 \le 4\;\|D^{-1}\|_2\;(1-\gamma)^{-1}\;\gamma^{n-2|k-i|},$$
$$|i-j| = |k-m|,\quad k+m = n+1,\quad 1\le i,j\le n.$$

Proof: For simplicity, assume that $i \le k$. (The argument for $i \ge k$ is similar.) Let $(A')^{-1}$ be a lower local inverse of A with $r_\ell = n-(k-i)$ and $(A'')^{-1}$ be an upper local inverse of A with $r_h = k-i$. By construction, both $A'$ and $A''$ are block $n-(k-i)\times n-(k-i)$ matrices and

$$([A']^{-1})_{i,j} = ([A'']^{-1})_{k-r_h,m-r_h}.$$



From Theorem 6.3,

$$\|([A']^{-1})_{i,j}-(A^{-1})_{i,j}\|_2 \le \|D^{-1}\|_2\;(1-\gamma)^{-1}\;\gamma^{1+2r_\ell-(i+j)}$$

and Corollary 6.4,

$$\|([A'']^{-1})_{k-r_h,m-r_h}-(A^{-1})_{i,j}\|_2 \le \|D^{-1}\|_2\;(1-\gamma)^{-1}\;\gamma^{i+j-2r_h-1}.$$

The exponents can be written as

$$\begin{aligned} 1+2r_\ell-(i+j) &= 1+2n-2k+2i-i-j \\ &= 1+2n-2k+k-m \\ &= n \end{aligned}$$

and

$$k+m-2r_h-1 = n-2(k-i).$$

The result follows from the triangle inequality and the simple relation

$$\gamma^n + \gamma^{n-2(k-i)} \le 2\,\gamma^{n-2(k-i)}.$$

Q.E.D.

## IV.7 Local Cholesky Factorizations of Matrices

If the matrix A is partitioned as in (6.2), then $M\,M^T = A''$ is a local Cholesky factorization of A. In this section we develop local error bounds for local Cholesky factorizations of symmetric, positive definite, block tridiagonal matrices. We also show that, for a matrix $A = L\,L^T$ having identical block rows, the block rows of Cholesky factor L are nearly identical. Consequently, for any fixed accuracy, there are only a fixed number of number of significantly different blocks in L and these blocks can be computed using a fixed number of operations.

The Cholesky factor L is a lower triangular, block bi-diagonal matrix with the same partitioning as A. Blocks $L_{i,i}$ and $L_{i,i-1}$ satisfy

$$D_1 = L_{1,1}L_{1,1}^T$$

$$(7.1)\quad D_i = L_{i,i}L_{i,i}^T + L_{i,i-1}L_{i,i-1}^T$$

$$C_{i-1}^T = L_{i,i-1}L_{i-1,i-1}^T, \qquad 2\le i \le n.$$

Thus,

$$L_{i,i}L_{i,i}^T = \begin{cases} D_1, & i = 1 \\ D_i - C_{i-1}^T(L_{i-1,i-1})^{-T}(L_{i-1,i-1})^{-1}C_{i-1}, & 2\le i \le n. \end{cases}$$

$$(7.2)$$

$$L_{i,i-1} = C_{i-1}^T L_{i-1,i-1}^{-T}, \qquad 2\le i \le n.$$

If $L[i-1]$ is the $(i-1)\times(i-1)$ principal block submatrix of L, then

$$L[i-1]_{i-1,i-1} = L_{i-1,i-1}, \quad 2\le i \le n.$$

The matrices $L[i-1]$ and $L[i-1]^{-1}$ are both lower triangular, so that

$$(L[i-1]^{-1})_{i-1,i-1} = (L[i-1]_{i-1,i-1})^{-1} \equiv (L_{i-1,i-1})^{-1}.$$

Moreover, because $A[i-1] = L[i-1]\,L[i-1]^T$,

$$A[i-1]^{-1} = L[i-1]^{-T}\,L[i-1]^{-1}.$$

Therefore, since $L[i-1]^{-T}$ is upper triangular and $L[i-1]^{-1}$ is lower triangular,

$$(7.3)\quad (A[i-1]^{-1})_{i-1,i-1} = (L[i-1]^{-T})_{i-1,i-1}\,(L[i-1]^{-1})_{i-1,i-1}$$

$$= (L[i-1]_{i-1,i-1})^{-T}\,(L[i-1]_{i-1,i-1})^{-1}$$

$$= (L_{i-1,i-1})^{-T}\,(L_{i-1,i-1})^{-1}, \qquad 2\le i \le n,$$



and from (7.2) we can obtain an expression for the products of the diagonal blocks of the Cholesky factor L

$$(7.4)\quad L_{i,i}L_{i,i}^T = \begin{cases} D_1, & i = 1, \\ D_i - C_{i-1}^T(A[i]^{-1})_{i-1,i-1}C_{i-1}, & 2\le i \le n. \end{cases}$$

By a similar argument, the products of the diagonal blocks of the local Cholesky factor M are given by

$$(7.5)\quad M_{i-r,i-r}M_{i-r,i-r}^T = \begin{cases} D_{r+1}, & i = r+2, \\ D_i - C_{i-1}^T(A''[i-r-1]^{-1})_{i-r-1,i-r-1}C_{i-1}, & r+2<i\le n. \end{cases}$$

Subtracting (7.5) from (7.4), we obtain

$$(7.6)\quad E_i = \begin{cases} C_{i-1}^T((A[i-1]^{-1})_{i-1,i-1} - (A''[i-r-1]^{-1})_{i-r-1,i-r-1})C_{i-1}, & i\ge r+2 \\ C_r^T\,(A[r]^{-1})_{r,r}\,C_r, & i=r+1. \end{cases}$$

where

$$E_i \equiv L_{i,i}L_{i,i}^T - M_{i-r,i-r}M_{i-r,i-r}^T, \qquad r+1\le i \le n.$$

In the following result, we apply Corollary 6.4 to bound $\| E_i \|_2$, $r+1\le i \le n$.

<u>LEMMA 7.1</u>

If $M\,M^T = A''$ is a local factorization of A, then

$$\| E_i \|_2 \le 2\,\kappa(D)\, \| D \|_2\, (1-\gamma)^{-1}\, \gamma^{2(i-r)}, \quad r+1\le i \le n.$$

<u>Proof:</u> From (7.6), the 2-norms of the error matrices satisfy

$$\| E_i \|_2 \le \| C_{i-1}^T \|_2^2\, \| (A[i-1]^{-1})_{i-1,i-1} - (A''[i-r-1]^{-1})_{i-r-1,i-r-1} \|_2, \quad i\ge r+2$$

$$\| E_{r+1} \|_2 \le \| C_r^T \|_2^2\, \| (A[i-1]^{-1})_{r,r} \|_2 .$$

Furthermore, from (5.8),

$$\| C_{i-1} \|_2 \leq \| D \|_2 \gamma^2, \quad r+1 \leq i \leq n.$$

Applying Corollary 6.4,

$$\| E_i \|_2 \leq \| D \|_2^2 \gamma^2 \, 2 \, \| D^{-1} \|_2 (1-\gamma)^{-1} \gamma^{2i-2r-1}, \quad r+2 \leq i \leq n,$$

and Theorem 3.5,

$$\| E_{r+1} \|_2 \leq \| D \|_2^2 \gamma^2 \, 2 \, \| D^{-1} \|_2 (1-\gamma)^{-1} \gamma^0,$$

we obtain the result.

Q.E.D.

Lemma 7.1 provides a bound on the differences of the products of the diagonal blocks of the local and full Cholesky factorizations. In Theorem 7.3 we obtain a bound on the differences of the blocks themselves by applying the following stability result for the Cholesky factorization [S9].

LEMMA 7.2 [S9]

Let B be an $m \times m$, symmetric, positive definite matrix and $R R^T$ be the Cholesky factorization of B. If H is an $m \times m$, symmetric matrix such that $B + H$ is positive definite and

$$\| H \|_2 \leq (2 \tau)^{-2}$$

where

$$\tau \equiv m \, \| R^{-1} \|_2 \, (1 + 2^{\frac{1}{2}} \kappa(R)) ,$$



then $B + H$ has the Cholesky factorization $(R')(R')^T$ and

$$\| R' - R \|_2 \leq 2 \tau \| H \|_2.$$

THEOREM 7.3

If $L L^T$ is the Cholesky factorization of A and $M M^T$ is a local Cholesky factorization of A, then

(7.7) $\| L_{i,i} - M_{i-r,i-r} \|_2 \leq 2\tau_A \Omega_A \gamma^{2(i-r)}, \qquad r+\mu_A \leq i \leq n,$

(7.8) $\| L_{i,i-1} - M_{i-r,i-r-1} \|_2 \leq 2\tau_A \Omega_A \| D \|_2 \gamma^{1+2(i-r)}, \quad r+\mu_A+1 \leq i \leq n.$

where

(7.9) $\Omega_A \equiv 2 \kappa(D) \, \| D \|_2 (1-\gamma)^{-1},$

(7.10) $\mu_A \equiv \max( 1, r - 2 \log_\gamma (4 \Omega_A \tau_A^2) ),$

and

(7.11) $\tau_A \equiv ( \max_{r+1 \leq i \leq n} \operatorname{ord}(D_i) ) \;\; \| A^{-1} \|_2^{\frac{1}{2}} (1 + 2^{\frac{1}{2}} \kappa^{\frac{1}{2}}(A)) .$

Proof: The constant $\mu_A$ defined in (7.10) is chosen so that the hypotheses of Lemma 7.1 are satisfied for the product-error matrices, i.e.,

$$\| E_i \| \leq \Omega_A \gamma^{2(i-r)} \leq (2 \tau^2(L_{i,i}))^{-2}, \quad i \geq \mu_A,$$

Consequently, we can obtain result (7.7) by applying Lemma 7.2 to the result of Lemma 7.1 with

$$B \equiv L_{i,i} L_{i,i}^T$$

and

$$H \equiv E_i \equiv M_{i-r-1,i-r-1} M_{i-r-1,i-r-1}^T - L_{i,i} L_{i,i}^T.$$

All that remains is to compute the constants $\tau$ and $\mu$ in terms of A.

From Lemma 7.2,

$$\tau(L_{i,i}) \equiv \mathrm{ord}(D_i)\ \| L_{i,i}^{-1} \|_2\ (1 + 2^{\frac{1}{2}}\kappa(L_{i,i})),$$

and from Lemma 3.4,

$$\tau(L_{i,i}) \le \{ \max_{r+1 \le i \le n} \mathrm{ord}(D_i) \}\ \| L^{-1} \|_2\ (1 + 2^{\frac{1}{2}}\kappa(L))$$

Because [S8,p.191]

$$\| L \|_2^2 = \| L L^T \|_2 = \| A \|_2$$

$$\| L^{-1} \|_2^2 = \| L^{-1} L^{-T} \|_2 = \| A^{-1} \|_2 ,$$

we can bound $\tau$ by

$$\tau \le \tau_A \equiv \{ \max_{r+1 \le i \le n} \mathrm{ord}(D_{i,i}) \}\ \| A^{-1} \|_2^{\frac{1}{2}} (1 + 2^{\frac{1}{2}}\kappa^{\frac{1}{2}}(A)).$$

To obtain result (7.8), we rewrite (7.2) as

$$L_{i,i-1} - M_{i-r,i-r-1} = C_{i-1}^T (L_{i-1,i-1}^{-1} - M_{i-r-1,i-r-1}^{-1}), \qquad r+2 \le i \le n.$$

Consequently,

$$\| L_{i,i-1} - M_{i-r,i-r-1} \|_2 \le \| C_{i-1}^T \|_2\ \| (L_{i-1,i-1}^{-1} - M_{i-r-1,i-r-1}^{-1}) \|_2$$

and the desired result follows from (7.7) and (5.8)

Q.E.D.

Frequently, as in (6.3), matrices are composed of identical block rows. For such matrices, the blocks $L_{i,i}$ and $L_{i,i-1}$ of the factorization converge to $L_{n,n}$ and $L_{n,n-1}$ as i increases. Consequently, if some fixed precision is required, only a fixed number of blocks of the factorization need to be computed, independent of the order of the matrix. A similar result has been derived by Malcolm and Palmer [M2] for the special case of point tridiagonal matrices.



COROLLARY 7.4

For a matrix A with identical rows,

$$\| L_{n,n} - L_{j,j} \|_2 \le 2\, \tau_A\, \Omega_A\, \gamma^{2j}, \qquad r+\mu_A \le j \le n$$

and

$$\| L_{n,n-1} - L_{j,j-1} \|_2 \le 2\, \tau_A\, \Omega_A\, \| D \|_2\, \gamma^{1+2j}, \qquad r+\mu_A+1 \le j \le n,$$

where $\tau_A$, $\Omega_A$, and $\mu_A$ are defined in (7.9), (7.10), and (7.11).

Proof: Fix $j \ge \mu_A$, let $r = n-j$, and partition A as in (6.2). Observe that

$$L_{j,j} = M_{n-r,n-r}$$

$$L_{j,j-1} = M_{n-r,n-r-1} .$$

Then apply Theorem 7.3 to obtain

$$\| L_{n,n} - L_{j,j} \|_2 = \| L_{n,n} - M_{n-r,n-r} \|_2 \le 2\, \tau_A\, \Omega_A\, \gamma^{2(n-(n-j))}$$

and

$$\| L_{n,n-1} - M_{n-r,n-r-1} \|_2 = \| L_{n,n-1} - L_{j,j-1} \|_2 \le 2\, \tau_A\, \Omega_A\, \| D \|_2\, \gamma^{1+2(n-(n-j))}.$$

Q.E.D.

## IV.8 Limitations and Generalizations

For a set of symmetric, positive definite, block tridiagonal matrices in which the condition number increases rapidly with n, e.g.,

$\kappa(A) > K\,n$, for some positive K independent of n,

the results of this chapter will be of limited value. In this case, the exponential damping bound

$$\| (A^{-1})_{i,j} \|_2 \leq \| D^{-1} \|_2 \,(1-\gamma)^{-1} \gamma^{|i-j|}$$

is essentially equivalent to the trivial bound

$$\| (A^{-1})_{i,j} \|_2 \leq \| D^{-1} \|_2 \,\kappa(A).$$

Many of the results of this chapter can be applied to more general classes of matrices. For example, a similar development applies to banded, diagonally dominant matrices. With some minor changes, the results also apply to block 2-cyclic matrices. The only major difference from the previous development is that the distance function $|i-j|$ is replaced by dist(i,j), the distance between the $i^{th}$ and $j^{th}$ nodes of the block graph of the matrix [V1,§3.3].

# Chapter V

# Local Dependence and Least-Squares Splines

## V.1 Introduction

Since the B-spline Gram matrices are symmetric, positive definite, banded, and well-conditioned, we can apply the local dependence theory of Chapter IV to develop an analogous theory of local dependence for least-squares splines. The results include an exponential damping bound for the inverse of the Gram matrix, simple local dependence bounds, error bounds for local solutions, and asymptotically optimal, local $L_\infty$ error bounds for least-squares spline approximation.

There has been considerable work concerning the local dependence of spline approximations. Early results include simple local dependence bounds for arbitrary-order spline interpolates with uniform knot spacing (Ahlberg, Nilson, and Walsh [A2]) and for cubic spline interpolates with arbitrary knot spacing (Kershaw [K3]). Both results were based on the explicit computation of the inverse of the spline interpolation matrix. Later, Kammerer, Reddien, and Varga [K1,K2] employed the Kershaw result to prove local convergence bounds for the cubic and quadratic spline interpolates. More recently, Liou [L2] duplicated the earlier proofs and implemented a local solution algorithm to solve large interpolation

problems in limited storage. Other local dependence results, though not explicitly identified as such, were developed by Schoenberg [S2] in studies of the asymptotic behavior of the cardinal spline basis functions.

Powell [P2] derived early local dependence bounds for least-squares cubic splines by studying the behavior of the solution to the seven-term recurrence relation leading to the elements of the inverse of the Gramian. Somewhat later, Domsta [D7] published exponential damping bounds for B-spline Gram matrices of arbitrary order with uniform knot spacing. Recently, Douglas, and Wahlbin [D8,D9], deBoor [B6], and Demko [D3,D4] developed exponential damping bounds for general B-spline Gram matrices and used these bounds to develop local, $L_\infty$ error bounds for least-squares splines. Eisenstat, Lewis, and Schultz [E2,E4] implemented local solution algorithms to solve least-squares problems of unlimited size using storage of fixed size.

Our derivation of the local dependence properties of least-squares splines is based on the local dependence properties of matrices developed in Chapter IV. In §2 we introduce the notation and basic results. We diagonally scale the normal equations and partition the Gramian as a block tridiagonal matrix of $(k-1)\times(k-1)$ blocks. We use the bounds on the condition number of the Gramian in §II.3 to derive an exponential damping bound on the elements of the inverse of the Gramian. In §3 and §4, we use this exponential damping bound and the results of Chapter IV to develop local dependence and local solution results for least-squares splines. These results lead to locally optimal $L_\infty$ error bounds (which are somewhat stronger than the earlier results in



[D3,D4]). In §5, we give several simple numerical examples for piecewise linear splines. In §6, we discuss some limitations and possible extensions of the results. In particular, we extend all of the results to the discrete least-squares approximation of functions sampled at data points.

## V.2 An Exponential Damping Bound

To simplify notation, we assume that the order of the B-spline Gram matrix is given by

$$n = \hat{n}\cdot\hat{k}, \qquad \hat{k} \equiv k-1,$$

for some positive integer $\hat{n}$, so that G can be partitioned as an $\hat{n}$ by $\hat{n}$ block tridiagonal matrix of $\hat{k}$ by $\hat{k}$ blocks. While not essential, this restriction on n greatly simplifies notation.

To apply the matrix local dependence results of Chapter IV to the partitioned Gram matrices, we need a bound on the damping factor $\gamma$ of (IV.3.2), independent of order. Because $\gamma$ is bounded by a monotonically increasing function of the condition number (see Theorem IV.3.3), it is sufficient to show that the $\ell_2$ condition number of the Gramian is bounded.

In general, the condition number of the Gramian cannot be bounded *independent of the number of knots, i.e., there exist knot vectors for* which the Gram matrix has an arbitrarily large condition number. For example, if $k = 1$ and $t_i = \beta^i$, $\beta < 1$, $1 \le i \le n+1$, then $\kappa_2(G) = \beta^{1-n}$. However, for an appropriate diagonal scaling, the condition number of

the scaled Gram matrix is bounded, independent of $\underline{t}$. One such scaling is the 2-scaling (II.2.19):

(2.1) $\hat{G} = E^{-\frac{1}{2}} G E^{-\frac{1}{2}}$

where

$$E \equiv \operatorname{diag}(e_1, e_2, \ldots, e_n),$$

(2.2)

$$e_i \equiv \| N_{i,k} \|_{L_1} = \frac{t_{i+k} - t_i}{k}, \quad 1 \leq i \leq n.$$

Corollary II.3.1 provides a bound on the eigenvalues of the diagonally-scaled Gram matrix $\hat{G}$. This bound and Theorem IV.3.3 lead to a bound on the damping constant (see (IV.3.2))

$$\gamma_k \equiv \rho(\hat{D}^{-1}\hat{C}),$$

where $\hat{D}$ is a matrix containing only the diagonal blocks of $\hat{G}$, and $\hat{C}$ is a matrix containing only the off-diagonal blocks.

<u>LEMMA 2.1</u>

For any knot vector $\underline{t}$,

$$\gamma_k \leq \frac{\Lambda_k^2 - 1}{\Lambda_k^2 + 1} < 1 - \Lambda_k^{-2} < 1,$$

$$(1-\gamma_k)^{-1} \leq \Lambda_k^2,$$

$$\| \hat{D} \|_2 \leq \| \hat{G} \|_2 \leq 1,$$

$$\| \hat{D}^{-1} \|_2 \leq \| \hat{G}^{-1} \|_2 \leq \Lambda_k^2,$$

where $\Lambda_k$ is defined in Theorem II.2.1.



<u>Proof:</u> From Theorem II.2. and the proof of Corollary II.3.1,

(2.3) $\| \hat{G}^{-1} \|_2^{-1} = \lambda_{min}(\hat{G}) \geq \Lambda_k^{-2}$

and

(2.4) $\| \hat{G} \|_2 = \lambda_{max}(\hat{G}) \leq 1$ .

Thus, the condition number of $\hat{G}$ satisfies

$$\kappa(\hat{G}) \leq \frac{\lambda_{max}(\hat{G})}{\lambda_{min}(\hat{G})} \leq \Lambda_k^2.$$

From Corollary IV.3.3 with $P = I$,

$$\gamma_k \leq \frac{\kappa(\hat{G}) - 1}{\kappa(\hat{G}) + 1} \leq \frac{\Lambda_k^2 - 1}{\Lambda_k^2 + 1} < 1 - \Lambda_k^{-2},$$

and we obtain the first two results.

*The matrix* $\hat{D}$ *contains only diagonal submatrices of* $\hat{G}$. Consequently, from (2.3), (2.4), and [S8,p.266,308],

$$\| \hat{D}^{-1} \|_2 \leq \| \hat{G}^{-1} \|_2 \leq \Lambda_k^2,$$

$$\| \hat{D} \|_2 \leq \| \hat{G} \|_2 \leq 1 .$$

Q.E.D.

The following "exponential damping" bound on the blocks of the inverse matrix $\hat{G}^{-1}$ follows directly from *Lemma 2.1 and* Theorem IV.3.5. Note that this bound does not depend on the number of knots or their relative spacing.

LEMMA 2.2

For any knot vector $\underline{t}$,

$$\| (\hat{G}^{-1})_{p,q} \|_2 \leq \Lambda_k^4 \; \gamma_k^{|p-q|}, \quad 1 \leq p,q \leq \hat{n},$$

where

$$\gamma_k < 1 - \Lambda_k^{-2}.$$

Because of the scaling matrices $E^{-\frac{1}{2}}$ in (2.1), the local dependence results for least-squares splines do not follow directly from Lemma 2.2 and the corresponding results of Chapter IV. It will be necessary to re-derive these results in the more general setting of the diagonally scaled Gram matrices.

The basis coefficient vector of the least-squares spline $P_{S(k,\underline{t})}f(t)$ is the solution to the diagonally scaled normal equations

$$E^{\frac{1}{2}} \hat{G} E^{\frac{1}{2}} \underline{a} = \underline{b} .$$

Consequently, the basis coefficient vector is given by

$$\underline{a} = E^{-\frac{1}{2}} \hat{G}^{-1} E^{-\frac{1}{2}} \underline{b} .$$

and the $i^{th}$ element of the basis coefficient vector satisfies

$$(2.5) \quad |a_i| \leq \sum_{j=1}^{n} e_i^{-\frac{1}{2}} |\hat{g}_{i,j}^{(-1)}| \, e_j^{-\frac{1}{2}} \, |b_j|, \quad 1 \leq i \leq n,$$



where $\hat{g}_{i,j}^{(-1)}$ is the i,j element of $\hat{G}^{-1}$. Since the elements of the right-hand side satisfy

$$(2.6) \quad |b_j| = \| N_{j,k} f \|_{L_1}$$
$$\leq \|f\|_\infty \, \| N_{j,k} \|_{L_1}$$
$$= \|f\|_\infty \, e_j, \qquad 1 \leq j \leq n,$$

we find that

$$|a_i| \leq \|f\|_\infty \sum_{j=1}^{n} \left(\frac{e_j}{e_i}\right)^{\frac{1}{2}} |\hat{g}_{i,j}^{(-1)}| , \qquad 1 \leq i \leq n.$$

Note that the role played by $(A^{-1})_{i,j}$ in the analysis of Chapter IV is now taken by

$$\left(\frac{e_j}{e_i}\right)^{\frac{1}{2}} |\hat{g}_{i,j}^{(-1)}| .$$

In the following result, we bound this new "exponential damping" term.

LEMMA 2.3

$$\left(\frac{e_j}{e_i}\right)^{\frac{1}{2}} |\hat{g}_{i,j}^{(-1)}| \leq \Lambda_k^4 \, \gamma_k^{-1} \, \alpha_k^{|i-j|}, \qquad 1 \leq i,j \leq n,$$

where the damping constant is given by

$$(2.7) \quad \alpha_k \equiv \sigma^{*} \gamma_k^{1/(k-1)}$$

and the local mesh ratio is defined as

$$(2.8)\quad \sigma \equiv \max_{1\le i\le n-1} \max\left(\frac{e_{i+1}}{e_i}, \frac{e_i}{e_{i+1}}\right)$$

$$= \max_{1\le i\le n-1} \max\left(\frac{t_{i+k+1}-t_{i+1}}{t_{i+k}-t_i}, \frac{t_{i+k}-t_i}{t_{i+k+1}-t_{i+1}}\right).$$

Proof: If $p = \left\lfloor\frac{i-1}{k-1}\right\rfloor$ and $q = \left\lfloor\frac{j-1}{k-1}\right\rfloor$, $1\le i,j\le n$, then from Lemma 2.2,

$$|\hat{g}^{(-1)}_{i,j}| \le \|(G^{-1})_{p,q}\|_2 \le \Lambda_k^4 \, \gamma_k^{|p-q|}.$$

Consequently, each element of $\hat{G}^{-1}$ satisfies

$$|\hat{g}^{(-1)}_{i,j}| \le \Lambda_k^4 \, \gamma_k^{\lfloor |i-j|/R\rfloor}$$

$$\le \Lambda_k^4 \, \gamma_k^{-1} (\gamma_k^{1/R})^{|i-j|/R}, \qquad 1\le i,j\le n;$$

and from (2.8), the mesh ratio satisfies

$$\left(\frac{e_j}{e_i}\right)^{\frac{1}{2}} \le \sigma^{|i-j|/2}.$$

Thus,

$$\left(\frac{e_j}{e_i}\right)^{\frac{1}{2}} |\hat{g}^{(-1)}_{i,j}| \le \Lambda_k^4 \, \gamma_k^{-1} \sigma^{|i-j|/2} \alpha_k^{|i-j|} = \Lambda_k^4 \, \gamma_k^{-1} \alpha_k^{|i-j|}, \quad 1\le i,j\le n.$$

Q.E.D.

Lemma 2.3 plays the same role in subsequent analysis as the "exponential damping" bound of Theorem IV.3.5. Note that this result is not particularly useful if $\alpha_k > 1$, since the exponential damping term increases with $|i-j|$ and sums of the form $\sum_{\ell=0}^{\infty} \alpha_k^{\ell}$ do not converge.



Consequently, for sufficiently high local mesh ratios $\sigma$, none of the following results will apply.

## V.3 Simple Local Dependence

In the following simple local dependence bound, we fix the knot vector $\underline{t}$ and bound the effects of perturbing the approximated function locally [cf. A2,P2,K1,K2]. Using this result, we can bound $\| P_{S(k,\underline{t})} \|_\infty$ and show that the $L_\infty$ error in least-squares approximation is asymptotically optimal [cf. D8,D9,B5,D3,D4].

Consider a perturbation $\delta(t)$ satisfying

$$\delta(t) \equiv 0, \quad t \in [t_k, t_r], \quad k\le r\le n+1.$$

Because the least-squares spline projection is linear,

$$P_{S(k,\underline{t})}(f+\delta) - P_{S(k,\underline{t})} f = P_{S(k,\underline{t})}\delta .$$

In the following result we show that the perturbation $P_{S(k,\underline{t})}\delta(t)$, $t_k\le t\le t_r$, is bounded by a decreasing exponential in the number of knots separating the evaluation point $t$ and the point $t_r$.

THEOREM 3.1 [cf. A2,P2,K1,K2]

If $\alpha_k < 1$ and

$$\delta(t) \equiv 0, \qquad t_k\le t < t_r, \quad k\le r\le n,$$

then for $t \in [t_i, t_{i+1}]$, $k \le i \le n$,

$$|P_{S(k,\underline{t})}\delta(t)| \le \begin{cases} \|\delta\|_\infty \Lambda_k^4 \gamma_k^{-1} (1-\alpha_k)^{-1} \alpha_k^{r-k+1-i}, & i \le r-k+1 \\ 2\|\delta\|_\infty \Lambda_k^4 \gamma_k^{-1} (1-\alpha_k)^{-1}, & \text{otherwise} \end{cases}.$$

Proof: Since the B-splines are local (II.2.13), the first r-k elements of the right hand side $\underline{b}$ vanish, i.e.,

$$b_j = 0 \quad \text{if} \quad 1 \le j \le r-k.$$

Thus, from (2.5) and (2.6) the elements of the basis coefficient vector of the least-squares spline satisfy

$$|a_v| \le \|\delta\|_\infty \sum_{j=r-k+1}^{n} \left(\frac{e_j}{e_v}\right)^{\frac{1}{2}} \hat{g}_{v,j}^{(-1)}, \qquad 1 \le v \le n.$$

From Lemma 2.2

$$(3.1) \quad |a_v| \le \|\delta\|_\infty \Lambda_k^4 \gamma_k^{-1} \sum_{j=r-k+1}^{n} \alpha_k^{|v-j|}, \qquad 1 \le v \le n.$$

Since (see (IV.4.7) and (IV.4.8))

$$\sum_{j=r-k+1}^{n} \alpha_k^{|j-v|} \le \begin{cases} (1-\alpha_k)^{-1} \alpha_k^{r-k+1-v}, & \text{if } v \le r-k+1 \\ 2(1-\alpha_k)^{-1} \end{cases}$$

and (see Theorem II.2.1 and (II.2.13-15))

$$|P_{S(k,\underline{t})}\delta(t)| \le \max_{i-k+1 \le v \le i} |a_v|, \quad t \in [t_i, t_{i+1}], \quad k \le i \le n.$$

we obtain the result

$$|P_{S(k,\underline{t})}\delta(t)| \le \|\delta\|_\infty \Lambda_k^4 \gamma_k^{-1} \begin{cases} (1-\alpha_k)^{-1} \alpha_k^{r-k+1-i}, & \text{if } i \le r-k+1 \\ 2(1-\alpha_k)^{-1}, & \text{otherwise} \end{cases}.$$

Q.E.D.



An immediate consequence of Theorem 3.1 is a bound on the $L_\infty$ norm of the least-squares spline projection operator

$$(3.2) \quad \|P_{S(k,\underline{t})}\|_\infty \equiv \max_{\|f\|_\infty = 1} \|P_{S(k,\underline{t})}f\|_\infty \le 2\Lambda_k^4 \gamma_k^{-1} (1-\alpha_k)^{-1}.$$

Using this bound, we can show that the $L_\infty$ error in least-squares spline approximation is asymptotically optimal.

COROLLARY 3.2 [c.f. B5,D8,D9]

If $\alpha_k < 1$ and $B_{S(k,\underline{t})}f$ is the best $L_\infty$ approximation to f(t) in $S(k,\underline{t})$, then

$$\|f - P_{S(k,\underline{t})}f\|_\infty \le \left(1 + 2\Lambda_k^4 \gamma_k^{-1} (1-\alpha_k)^{-1}\right) \|f - B_{S(k,\underline{t})}f\|_\infty.$$

Proof: Since $P_{S(k,\underline{t})}$ reproduces splines in $S(k,\underline{t})$,

$$\begin{aligned} \|f - P_{S(k,\underline{t})}f\|_\infty &= \|f - B_{S(k,\underline{t})}f - P_{S(k,\underline{t})}(f - B_{S(k,\underline{t})}f)\|_\infty \\ &\le \|f - B_{S(k,\underline{t})}f\|_\infty + \|P_{S(k,\underline{t})}(f - B_{S(k,\underline{t})}f)\|_\infty \\ &\le \|f - B_{S(k,\underline{t})}f\|_\infty + \|P_{S(k,\underline{t})}\|_\infty \|f - B_{S(k,\underline{t})}f\|_\infty \\ &\le \left(1 + \|P_{S(k,\underline{t})}\|_\infty\right) \|f - B_{S(k,\underline{t})}f\|_\infty. \end{aligned}$$

The result follows from (3.2) and Theorem 3.1.

Q.E.D.

In particular, we can apply the $L_\infty$ error bounds of de Boor [B10] to obtain optimal order $L_\infty$ error bounds for least-squares spline approximation.

COROLLARY 3.3 [B10,B6]

If $\alpha_k < 1$ and $f \in C^{k-1}[t_k, t_{n+1}]$ then

$$\| D^\ell (f - P_{S(k,\underline{t})} f) \|_\infty \le K(k,\sigma,\alpha_k)\ |\underline{t}|^{k-1-\ell}\ \omega(D^{k-1} f, |\underline{t}|), \quad 0 \le \ell \le k-1,$$

where $K(k,\sigma,\alpha_k)$ is a constant, the mesh width is defined by

$$|\underline{t}| \equiv \max_{1 \le i \le n} t_{i+k} - t_i ,$$

and the modulus of continuity is defined by

$$\omega(g, h) \equiv \max_{x,x+h \in [t_k, t_{n+1}]} \left| \frac{g(x+h) - g(x)}{h} \right| .$$

V.4 Local Solution

Let

$$\underline{t}^{<r>} \equiv ( t_1^{<r>}, \ldots, t_{r+2k-1}^{<r>} ), \quad r \ge 0,$$

be a B-rpline knot vector. If $\underline{t}$ is another B-spline knot vector with



the same initial $r+2k-1$ knots, i.e.,

$$\underline{t}[r+2k-1] \equiv ( t_1, \ldots, t_{r+2k-1} ) = \underline{t}^{<r>},$$

then $P_{S(k,\underline{t}^{<r>})} f$ is a local least-squares spline for the knot vector $\underline{t}$. In this section, we show that the local least-squares spline $P_{S(k,\underline{t}^{<r>})} f$ is close to the least-squares spline $P_{S(k,\underline{t})} f$. Moreover, we find that the local $L_\infty$ error in least-squares spline approximation is within a constant of that for local best $L_\infty$ spline approximation.

Because the B-splines are local (see (II.2.13)), the principal $r \times r$ submatrices of the Gram matrices for $\underline{t}^{<r>}$ and $\underline{t}$ are identical, i.e.,

$$G_{\underline{t}^{<r>}}[r] = G_{\underline{t}}[r] \equiv G^{<r>}.$$

Similarly, the first r entries in the right-hand sides are identical, i.e.,

$$\underline{b}_{\underline{t}^{<r>}}[r] = \underline{b}_{\underline{t}}[r] \equiv \underline{b}^{<r>}.$$

The local solution vector $\underline{a}^{<r>}$ is defined to be the solution to

$$(4.1) \quad G^{<r>} \underline{a}^{<r>} = \underline{b}^{<r>}$$

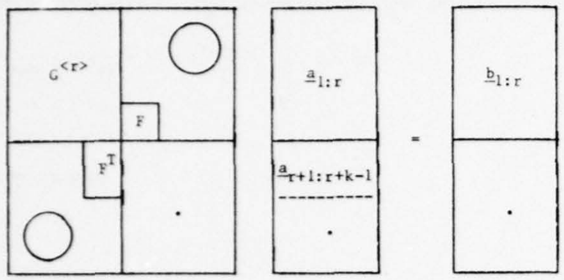
and the local solution spline is in turn given by

$$s^{<r>}(t) = \sum_{i=1}^{r} a_i^{<r>} N_{i,k}(t) .$$

Note that this local solution spline is not a least-squares spline, but as we will show later, it is close to both the least-squares spline $P_{S(k,\underline{t})} f$ and the local least-squares spline $P_{S(k,\underline{t}^{<r>})} f$.

The basis coefficient vector of the least-squares spline $P_{S(k,\underline{t})}f$ is the solution to the linear system

(4.2)




where F is the lower-triangular submatrix

$$F \equiv \begin{bmatrix} g_{r-k+1,r+1} & & & \\ g_{r-k+2,r+1} & g_{r-k+2,r+2} & & \bigcirc \\ \cdots & \cdots & & \\ g_{r-1,r+1} & \cdots & g_{r-1,r+k-2} & \\ g_{r,r+1} & \cdots & g_{r,r+k-2} & g_{r,r+k-1} \end{bmatrix}$$

and the partial vectors $\underline{a}_{u:v}$ are defined by

$$\underline{a}_{u:v} = (\, a_u,\ a_{u+1},\ \ldots,\ a_v \,), \qquad 1 \le u \le v \le n.$$

From (4.1), (4.2), and §IV.5, the difference between the basis coefficient vectors of the local solution spline and the least-squares spline is given by

$$(4.3)\quad \underline{a}_{1:r} - \underline{a}^{<r>} = (G^r)^{-1} \begin{bmatrix} \bigcirc \\ \hline F\ \underline{a}_{r+1:r+k-1} \end{bmatrix}.$$



In the following theorem, we apply Lemma 2.3 to bound the difference between the local solution spline and the least-squares spline at any point $t \in [t_k, t_{r-k+2}]$.

THEOREM 4.1

If $\alpha_k < 1$, then for $t \in [t_i, t_{i+1}]$, $i \le r-k+1$,

$$|s^{<r>}(t) - P_{S(k,\underline{t})}f(t)| \le 2k^2 \ \|f\|_\infty\ \gamma_k^{-3}\ \Lambda_k^8\ (1-\alpha_k)^{-1}\ \alpha_k^{r-i},$$

and

$$|P_{S(k,\underline{t}^{<r>})}f(t) - P_{S(k,\underline{t})}f(t)| \le 4k^2 \ \|f\|_\infty\ \gamma_k^{-3}\ \Lambda_k^8\ (1-\alpha_k)^{-1}\ \alpha_k^{r-i},$$

Proof: From (4.3) and (2.1), with $1 \le v \le r$,

$$|a_v^{<r>} - a_v| \le \sum_{j=r-k+1}^{r} |g_{v,j}^{<-r>}| \sum_{\ell=r+1}^{r+k-1} |g_{j,\ell}^{<r>}|\ |a_\ell|$$

$$\le \|\underline{a}\|_\infty \sum_{j=r-k+1}^{r} \sum_{\ell=r+1}^{r+k-1} \left(\frac{e_j}{e_v}\right)^{\frac{1}{2}} |\hat{g}_{v,j}^{<-r>}| \left(\frac{e_\ell}{e_j}\right)^{\frac{1}{2}} |\hat{g}_{j,\ell}^{<r>}|.$$

We will bound each of the four terms separately. From (3.1) and (IV.4.7),

$$\|\,\underline{a}\,\|_\infty \le 2\ \Lambda_k^4\ \gamma_k^{-1}\ (1-\alpha_k)^{-1}\ \|f\|_\infty\ ;$$

from (2.8),

$$\max_{r-k+1 \le j \le r}\ \max_{r+1 \le \ell \le r+k-1} \left(\frac{e_\ell}{e_j}\right)^{\frac{1}{2}} \le \sigma^k\ ;$$

from Lemma 2.3,

$$\left(\frac{e_j}{e_v}\right)^{\frac{1}{2}} |\hat{g}^{<-r>}_{\ell,j}| \leq \Lambda_k^4 \gamma_k^{-1} \alpha_k^{|j-v|} ;$$

and applying [S8,p.183] and Lemma 2.1,

$$|\hat{g}^{<r>}_{j,\ell}| \leq \| \hat{G}^{<r>} \|_2 \leq \| \hat{G} \|_2 \leq 1.$$

Thus,

$$|a_v^{<r>} - a_v| \leq 2k^2 \sigma^{R} \gamma_k^{-2} \Lambda_k^8 (1-\alpha_k)^{-1} \alpha_k^{r-k+1-v}, \quad 1 \leq v \leq r-k+1,$$

and the result follows from the inequality (*see* Theorem II.2.1 and (II.2.13-15))

$$|s^{<r>}(t)| \leq \max_{i-k+1 \leq v \leq i} |a_v^{<r>} - a_v| , \qquad t \in [t_i, t_{i+1}].$$

The second result follows from the first result and the triangle inequality.

Q.E.D.

An immmediate consequence of Theorem 4.1 is a bound on the local approximation error in the least-squares spline in terms of the error in the local least-squares spline.



COROLLARY 4.2

If $\alpha_k < 1$, then for $t \in [t_i, t_{i+1}]$, $i \leq r-k+1$,

$$|f(t) - P_{S(k,\underline{t})} f(t)| \leq | f(t) - P_{S(k,\underline{t}^{<r>})} f(t) |$$
$$+ 4k^2 \| f \|_\infty \gamma_k^{-3} \Lambda_k^8 (1-\alpha_k)^{-1} \alpha_k^{r-i} .$$

From Corollary 3.2, the $L_\infty$ *error in the local least-squares* spline $P_{S(k,\underline{t}^{<r>})} f(t)$ is within a constant of local best $L_\infty$ approximation. Thus, for r-i sufficiently large, the exponentially decaying term in the result of Corollary 4.2 is neglible and the local $L_\infty$ error in least-squares spline approximation is within a constant of that for local best $L_\infty$ approximation.

COROLLARY 4.3

If $\alpha_k < 1$ and $B_{S(k,\underline{t}^{<r>})} f$ is the local best $L_\infty$ approximation, then for $t \in [t_i, t_{i+1}]$, $i \leq r-k+1$,

$$|f(t) - P_{S(k,\underline{t})} f(t)| \leq \left(1 + 2\Lambda_k^4 \gamma_k^{-1} (1-\alpha_k)^{-1}\right) |f(t) - B_{S(k,\underline{t}^{<r>})} f(t)|$$
$$+ 4k^2 \| f \|_\infty \gamma_k^{-3} \Lambda_k^8 (1-\alpha_k)^{-1} \alpha_k^{r-i} .$$

The first term of the previous result, the *local approximation error*, can be bounded using Corollary 3.3. If r-i is sufficiently large, then the local $L_\infty$ error bound depends only on local properties of the function and the knots. Moreover, if the local approximation error term vanishes, then an exponential error bound can be obtained.

<u>COROLLARY 4.4</u>

If $\alpha_k < 1$ and $f \in C^{k-1}[t_k, t_{r+k}]$, then for $0 \le \ell \le k-1$ and $k \le i \le r-k+1$,

$$\| f - P_{S(k,\underline{t})} f \|_{L_\infty[t_k, t_{i+1}]} \le K_r(k,\sigma,\alpha_k)\, |\underline{t}^{<r>}|^{k-1-\ell}\, \omega_r(D^{k-1}f, |\underline{t}^{<r>}|)$$

$$+\ 4k^2\, \|f\|_\infty\, \gamma_k^{-3}\, \Lambda_k^8\, (1-\alpha_k)^{-1}\, \alpha_k^{r-i},$$

where $K_r(k,\sigma,\alpha_k)$ is a constant and the local modulus of continuity is defined by

$$\omega_r(g,h) \equiv \max_{x,x+h \in [t_k, t_{r+k}]} \left| \frac{g(x+h) - g(x)}{h} \right|.$$



V.5 <u>Numerical Examples</u>

In this section, we illustrate some applications and limitations of the results by applying them to simple examples involving piecewise linear splines. For integer knots, the piecewise linear B-spline Gram matrix is given by

$$(5.1)\quad G = \frac{1}{6}\begin{bmatrix} 2 & 1 & & & \bigcirc \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ \bigcirc & & & 1 & 2 \end{bmatrix}_{n\times n}.$$

The elements of the inverse of this matrix satisfy the bound of Lemma 2.3

$$|g_{i,j}^{-1}| \le \Lambda_k^4\, \gamma_k^{-1}\, \alpha_k^{|i-j|}, \quad 1 \le i,j \le n.$$

Since $\Lambda_2 = 2.5$ (see Theorem II.2.1), we conclude from Lemma 2.1 that

$$\alpha_2 = \gamma_2 \le \frac{2.5^2 - 1}{2.5^2 + 1} \approx .724$$

and

$$|g_{i,j}^{-1}| \le 54\,(.724)^{|i-j|}, \quad 1 \le i,j \le n.$$

This bound is not particularly sharp. The inverse of the piecewise linear Gram matrix (5.1) can be computed explicitly [G2,p.48], i.e.,

$$g_{i,j}^{-1} \approx \sqrt{3}\,(-1)^{|i-j|}\,(2-\sqrt{3})^{|i-j|}, \quad |i-j| \ll n, \quad n \gg 1$$

and

$$|g_{i,j}^{-1}| \le 8\,(2-\sqrt{3})^{|i-j|}, \quad 1 \le i,j \le n.$$

The actual damping constant $2-\sqrt{3} = .268$ is considerably smaller than our upper bound for $\alpha_2$.

Because the Gram matrix satisfies this exponential damping bound, distant function values have very little effect on the value of the least-squares spline at any one point. Consider the piecewise linear least-squares approximation of data from the step function (see Figure 5.2)

$$f(t) = \begin{cases} 1, & \text{if } t \geq 0, \\ 0, & \text{if } t < 0, \end{cases}$$

with the knots

$$\underline{t} = (-10,-10,-9, \ldots, 9, 10, 10).$$

In the left half-interval, the least-squares spline is essentially zero; and in the right half-interval, the least-squares spline is essentially one. Near the discontinuity at $t = 0$, the least-squares spline oscillates somewhat, but these oscillations decay rapidly in both directions.

In particular, from Corollary 3.2, if $t \in [t_i, t_{i+1}]$, $-10 \leq i \leq -1$, then

$$\begin{aligned} |P_{S(2,\underline{t})} f(t)| &\leq \Lambda_2^4 \alpha_2^{-k} \gamma_2^{-1} (1-\alpha_2)^{-1} \alpha_2^{1-i} \\ &\leq (2.5)^4 (.724)^{-2} (1-.724)^{-1} (.724)^{1-i} \\ &= (2.5)^4 (.724)^{-2} (1-.724)^{-1} (.724)^{1-i} \\ &\leq 270.4 \; (.724)^{1-i} . \end{aligned}$$



Otherwise,

$$\begin{aligned} |P_{S(k,\underline{t})} f(t)| &\leq 2 \Lambda_2^4 \gamma_2^{-1} (1-\alpha_2)^{-1} \\ &\leq (2.5)^4 (.724)^{-1} (1-.724)^{-1} \\ &\leq 196.7 . \end{aligned}$$

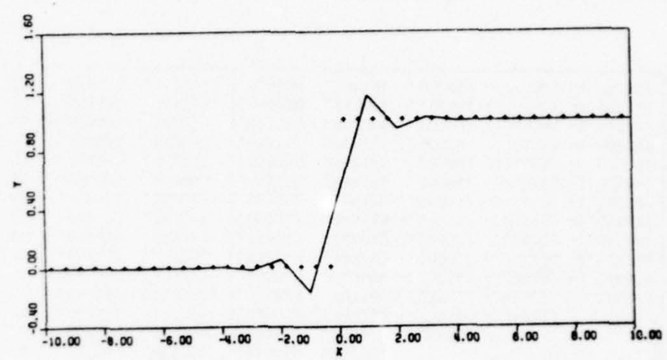
Again, these bounds are somewhat pessimistic, both in terms of the leading constant and the damping factor.

FIGURE 5.2

Local Dependence: Piecewise-Linear Least-Squares Approximation of the Step Function

$$f(t) = \begin{cases} 1 & \text{if } t \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

with $\underline{t} = (-10,-10,-9, \ldots, 9, 10,10)$



In many least-squares spline problems, we may wish to evaluate the least-squares spline in some interval $[t_k,t_{r+1}] \subseteq [t_k,t_{n+1}]$ without solving the entire problem. Such a situation might arise in data-fitting on a small computer without sufficient memory to store the entire Gram matrix, or in the on-line approximation of data where a partial solution is desired before the entire set of data is available (see Chapter VI and [E2,L2,E4]).

The local least-squares spline $P_{S(k,\underline{t}^{<r>})} f$ (see §4) is a good estimate for the least squares spline in the interval $[t_k,t_{r+1}]$. The error in the local least-squares spline decreases exponentially with the number of knots separating the evaluation point $t \in [t_k,t_{r-k+2}]$ and the point $t_r$. In the piecewise linear case the error is bounded by $(2160.)(.724)^{r-i}$ (see Theorem 4.1 and Table 5.1).



TABLE 5.1

Local Solution: Basis coefficients of Local Least-Squares Piecewise-Linear Splines for $n = 33$ and $f(t) = \cos(8\pi t)$ on $[0,1]$

| r | Basis Coefficients 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 2 | .196332 | .366298 | | | | | |
| 3 | -.061737 | .882437 | .562076 | | | | |
| 4 | .021604 | .715754 | 1.145466 | .401784 | | | |
| 5 | .008482 | .742000 | 1.053606 | .742978 | .004567 | | |
| 6 | .007472 | .744019 | 1.046538 | .769232 | -.093380 | -.395713 | |
| 7 | .008797 | .741370 | 1.055811 | .734789 | .035121 | -.675271 | -.564123 |
| 8 | .008367 | .742229 | 1.052805 | .745956 | -.006541 | -.719792 | -1.144379 |
| 9 | .008435 | .742093 | 1.053278 | .744197 | .000020 | -.744278 | -1.052996 |
| 10 | .008440 | .742083 | 1.053315 | .744062 | .000525 | -.746162 | -1.045964 |
| 11 | .008433 | .742097 | 1.053267 | .744239 | -.000137 | -.743690 | -1.055189 |
| 12 | .008435 | .742092 | 1.053282 | .744182 | .000077 | -.744491 | -1.052198 |
| 13 | .008435 | .742093 | 1.053280 | .744191 | .000044 | -.744365 | -1.052669 |
| 14 | .008435 | .742093 | 1.053280 | .744192 | .000041 | -.744355 | -1.052706 |
| 15 | .008435 | .742093 | 1.053280 | .744191 | .000044 | -.744368 | -1.052658 |
| 16 | .008435 | .742093 | 1.053280 | .744191 | .000043 | -.744364 | -1.052674 |
| 17 | .008435 | .742093 | 1.053280 | .744191 | .000044 | -.744365 | -1.052671 |
| 18 | .008435 | .742093 | 1.053280 | .744191 | .000044 | -.744365 | -1.052671 |

Suppose that the approximated function is very smooth or the mesh is very fine in some interval $[t_k,t_{r+2k-1}] \subseteq [t_k,t_{n+1}]$. Then, from the local solution error bound of Corollary 4.4, the $L_\infty$ error in both the local and global least-squares projection is small in the interval $[t_k,t_{r+1}]$.

This result is demonstrated by the least-squares piecewise-linear approximation of the function $f(t) = t_+^2$ on the interval $[-1,+1]$. In the subinterval $[1/2,1]$, where $f(t) = t^2$, the error decreases asymptotically as $n^{-2}$ (see Table 5.2a), the expected quadratic rate of convergence for piecewise-linear splines. However, in the interval $[-1,-1/2]$, where $f(t) \equiv 0$ and the local approximation error vanishes, the error decreases

exponentially (see Table 5.2b).

TABLE 5.2a

The Local Rate of Convergence of $P_{S(2,\underline{t})}f$
to $f(t) = t_+^2$ for n Uniformly Spaced Knots on [-1,1]

| n | $L_\infty$ Error in [1/2,1] | Rate |
|---|---|---|
| 3 | .2083333270 | |
| 7 | .0188746600 | 2.8341 |
| 11 | .0066759137 | 2.2994 |
| 15 | .0034017144 | 2.1738 |
| 19 | .0020576405 | 2.1267 |
| 23 | .0013774746 | 2.1005 |
| 27 | .0009861929 | 2.0840 |
| 31 | .0007408485 | 2.0706 |

TABLE 5.2b

The Local Rate of Convergence of $P_{S(2,\underline{t})}f$
to $f(t) = t_+^2$ for n Uniformly Spaced Knots in [-1,1]

| n | $L_\infty$ Error in [-1,-1/2] | Decay constant |
|---|---|---|
| 3 | 0.0416666667 | |
| 7 | 0.0008903134 | .382 |
| 11 | 0.0000874770 | .560 |
| 15 | 0.0000119742 | .608 |
| 19 | 0.0000019411 | .634 |
| 23 | 0.0000003482 | .651 |
| 27 | 0.0000000668 | .662 |
| 31 | 0.0000000134 | .670 |



## V.6 Limitations and Generalizations

While these results have been derived for least-squares polynomial spline approximation, the same results are valid for least-squares approximation with any splines for which a "B-spline" or local-support basis can be constructed. For example, such bases can be constructed for L-splines [J2,S4,S6], the set of piecewise smooth solutions to the differential equation $L^*L\,s = 0$, where L is a linear differential operator. Even-order polynomial splines are a special case of L-splines with $L = D^{k/2}$. For such bases, a compactness argument similar to that of Descloux [D5] and Fried [F5] yields a bound on the spectral condition number of the Gram matrix, independent of the number of knots, and the development of this chapter applies.

The results of this chapter also hold for the discrete least-squares approximation of functions. Assume that the abscissas and weights satisfy the hypotheses of Theorem II.4.3 and the ordinates are given by

$$y_\ell = f(x_\ell), \quad 1 \leq \ell \leq N.$$

Consequently, if we use the diagonal scaling matrix $\hat{E}$ of (II.5.1), then Corollary II.5.3 provides a bound on the condition number of the discrete Gramian and the remainder of the results follow as stated, with the substitution of $\Gamma_Q$ for $\Lambda_k$, sums for integrals, and the $X_\infty$ norm for the $L_\infty$ norm. To obtain spline regression error bounds in the $L_\infty$ norm, we observe that the $X_\infty$ norm is equivalent to the $L_\infty$ norm over splines (see Corollary II.5.5) so that the $X_\infty$ error bounds for spline regression

can be rewritten as $L_\infty$ error bounds with a minor adjustment of constants.

As we have already seen in §5, the bounds of this chapter are somewhat weak in terms of the leading constants and the damping factor $\alpha_k$. In Table 6.1, the damping constants $\alpha_k$ computed from (2.7) are compared with those computed numerically from the "empirical damping constant"

$$\alpha_{num} = \lim_{n\to\infty} \frac{\epsilon_n}{\epsilon_{n-1}} ,$$

where

$$\epsilon_n \equiv \| f(t) - P_{S(k,\underline{t})} f(t) \|_{L_\infty[t_k,t_{k+1}]}$$

for n+k uniformly spaced knots in [-1,1] and the function

$$f(t) = \begin{cases} 1 & \text{for } t \ \ [t_n,t_{n+1}] \\ 0 & \text{otherwise} \end{cases} .$$

TABLE 6.1
Damping Constants for Uniform Knot Spacing

| $k$ | $\alpha_{num}$ | $\alpha_k$ |
|---|---|---|
| 2 | .268 | .500 |
| 3 | .431 | .783 |
| 4 | .535 | .904 |
| 5 | .608 | .958 |
| 6 | .661 | .981 |



Another weakness of these results is that the bounds on the damping constant $\alpha_k$ exceed unity for meshes with a sufficiently large local mesh ratio. In fact, numerical experiments indicate that the damping constant $\alpha_{num}$ is bounded by a constant less than unity, even if the knot distribution is highly non-uniform. For example, consider the mesh

$$\begin{aligned} &t_1 = 0 \\ (6.1)\quad & \\ &t_i = \beta^i + t_{i-1}, \quad 0< \beta \le 1, \ 2\le i \le n+k, \end{aligned}$$

with the local mesh ratio

$$\sigma = \beta^{-1}.$$

In Table 6.2, we show the results of a numerical experiment in which $\alpha_{num}$ is computed for various values of $\beta$. While the damping constant is larger for highly nonuniform meshes (smaller $\beta$), for each value of k the damping constant remains smaller than $\frac{k-1}{k}$. This and other numerical experiments suggest that $\alpha_k$ may very well be bounded by $\frac{k-1}{k}$ for all knot vectors.

TABLE 6.2

Damping Constants for $\beta^i$ Meshes

| | $\beta$ | $\alpha_{num}$ |
|---|---|---|
| k = 2 | 1. | .268 |
| | .5 | .354 |
| | .2 | .433 |
| | .1 | .464 |
| | .01 | .496 |
| | .001 | .500 |
| k = 3 | 1. | .431 |
| | .5 | .549 |
| | .2 | .621 |
| | .1 | .644 |
| | .01 | .666 |
| k = 4 | 1. | .535 |
| | .5 | .663 |
| | .2 | .719 |
| | .1 | .735 |
| | .01 | .748 |
| k = 5 | 1. | .608 |
| | .5 | .728 |
| | .2 | .778 |
| | .1 | .789 |
| | .01 | .799 |
| k = 6 | 1. | .661 |
| | .5 | .784 |
| | .2 | .817 |
| | .1 | .825 |



For the special case of piecewise linear splines, the damping constant can be computed explicitly and bounded independent of $\underline{t}$. Here we will employ a somewhat different scaling of the normal equations,

$$\mathfrak{G}\,\underline{a} = \underline{\mathfrak{b}},$$

where

$$\mathfrak{G} = C\,G$$

$$\underline{\mathfrak{b}} = C\,\underline{b}$$

with

$$C \equiv \operatorname{diag}(\,c_1,\ c_2,\ \ldots,\ c_n\,), \qquad c_i = (t_{i+2}-t_i)^{-1}, \quad 1 \le i \le n.$$

With this scaling

$$(6.3)\quad \mathfrak{G} = \frac{1}{6}\begin{bmatrix} 2 & 1-\omega_1 & & & \bigcirc \\ \omega_2 & 2 & 1-\omega_2 & & \\ & \omega_3 & 2 & 1-\omega_3 & \\ \bigcirc & & & \cdots & \\ & & & & \cdots \end{bmatrix},$$

with

$$\omega_i = \frac{t_{i+1}-t_i}{t_{i+2}-t_i}, \qquad 1 \le i \le n.$$

Since [K3],

$$(6.4)\quad |\mathfrak{G}^{-1}_{i,j}| \le 3\left(\tfrac{1}{2}\right)^{|i-j|}, \qquad 2 \le i,j \le n-1.$$

and

$$|\mathfrak{b}_i| \le \|f\|_\infty, \qquad 1 \le i \le n,$$

we can employ (6.4) in place of Lemma 2.3 and derive all of the local dependence results for piecewise linear splines, independent of $\underline{t}$. Unfortunately, this argument does not generalize easily to higher-order splines.

# Chapter VI

# A Real-Time Algorithm

## VI.1 Introduction

In data-smoothing and signal-processing applications where large quantities of data are processed on-line, it is desirable to compute least-squares splines in real time, i.e., to compute the spline approximations while the data are being acquired. *In this chapter,* we apply the algorithm analysis of Chapter III, the matrix local dependence results of Chapter IV, and the least-squares spline local dependence results of Chapter V to develop a real-time, least-squares spline algorithm.

The algorithm scans the data only once, producing the B-spline coefficients for the least-squares spline as the data are scanned. For a fixed relative accuracy, the algorithm requires a fixed amount of storage (a few hundred locations for $10^{-7}$ relative accuracy). In general, it requires approximately the same total computation time as the serial algorithms of Chapter III, and in the special case of uniformly spaced knots and data, it requires significantly less computation time (4-12 multiplications per data point for cubic splines).

Real-time algorithms are easily implemented for strictly local approximation schemes such as simple digital filters [S8,§8], Hermite spline interpolates [A3,E6,M4], and discontinuous least-squares splines [P1,R3]. Only recently have limited-storage algorithms been introduced for smooth cubic spline interpolation [L2], continuous least-squares splines [I1], and smooth least-squares splines (see [E2;E4] for earlier versions of the algorithm presented here).

The algorithm is organized as a three process pipeline. In Process 1, the Gram matrix G and right-hand side $\underline{b}$ are computed. Because the completed rows of the Gram matrix and right-hand side are passed on to Process 2, only k (partially completed) rows of G and b are stored at any time. In *Process* 2, the factorization $G = L\ D\ L^T$ and the forward-solution $\underline{c} = D^{-1}L^{-1}\underline{b}$ are computed. Because the finished rows of L, D, and $\underline{c}$ are passed on to Process 3, only k rows of L, D, and $\underline{c}$ are stored at any time. In Process 3, accurate estimates for the basis coefficients of the least-squares spline are computed from local solutions to the normal equations. For some constant $\xi$ depending on the required accuracy, as few as $\xi+1$ rows of the factorization and forward-solution might be stored. The completed basis coefficient values are passed on to the output. The entire algorithm can be viewed as a least-squares spline "filter" which accepts a stream of data and produces a (slightly delayed) stream of B-spline coefficients.

The first two processes, which are described in §2 and §3, are pipelined, band-storage versions of Algorithm III.2.1 and Algorithm III.3.1. The third process, which is described in §4, is an



approximate, local back-solution. In §5, detailed operation counts are presented, and a number of implementation details are explored.

## VI.2 Process 1: Forming the Normal Equations

Process 1 is a straightforward implementation of Algorithm III.2.1 using Algorithm III.4.2 for interval location and Algorithm III.5.2 for B-spline evaluation. It accepts input data, forms rows of the Gram matrix G, computes elements of the vector $\underline{b}$, and passes the results to Process 2.

Since each data point affects only a k×k diagonal submatrix of G and k rows of $\underline{b}$, only those portions of G and $\underline{b}$ need to be stored (see Figure 2.1). These k rows of $\underline{b}$ *and* the lower triangle of G are stored in the arrays g[k,k] and b[k]. For example, with k = 4, during the processing of data lying in $[t_r, t_{r+1}]$, the arrays contain rows r-k+1 through r of G and $\underline{b}$

| Row | g[k,k] | | | | b[k] |
|---|---|---|---|---|---|
| 1 | $g_{r-3,r-6}$ | $g_{r-3,r-5}$ | $g_{r-3,r-4}$ | $g'_{r-3,r-3}$ | $b'_{r-3}$ |
| 2 | $g_{r-2,r-5}$ | $g_{r-2,r-4}$ | $g'_{r-2,r-3}$ | $g'_{r-2,r-2}$ | $b'_{r-2}$ |
| 3 | $g_{r-1,r-4}$ | $g'_{r-1,r-3}$ | $g'_{r-1,r-2}$ | $g'_{r-1,r-1}$ | $b'_{r-1}$ |
| 4 | $g'_{r,r-3}$ | $g'_{r,r-2}$ | $g'_{r,r-1}$ | $g'_{r,r}$ | $b'_r$ . |

Here g[i,j] and b[i] denote the elements of the arrays storing G and $\underline{b}$; $g_{i,j}$ and $b_i$ denote the values of elements of G and $\underline{b}$ stored in the arrays; and $g'_{i,j}$ and $b'_i$ denote the corresponding partially computed values.

FIGURE 2.1
Forming the Gram matrix in Process 1:
Processing data in $[t_r, t_{r+1}]$ for k = 4.

```
    The Band of the Lower Triangle of G     The Vector b
Row
  .     .                                        .
  .       .                                      .
  .         .                                    .
  .       X X X X                                X
  .         X X X X                              X
  .           X X X X                            X
r-k+1           + + + *                          *
  .               + + * *                        *
  .                 + * * *                      *
  r                   * * * *                    *
  .                     Y Y Y Y                  Y
  .                       Y Y Y Y                Y
  .                         Y Y Y Y              Y
```

X -- already computed and passed to Process 2
+ -- already computed but not yet passed to Process 2
* -- affected by data points in $[t_r, t_{r+1}]$

Y -- not yet computed

Only elements marked + or * are stored in Process 1.

When the computations for the first row of g[k,k] and b[k] are completed, i.e., when a data point not in the interval $[t_r, t_{r+1}]$ is acquired, that row is passed to Process 2; r is incremented; and the contents of the arrays are shifted to prepare for processing data in the interval $[t_{r+1}, t_{r+2}]$. Then the arrays contain

| Row | g[k,k] | | | | b[k] |
|---|---|---|---|---|---|
| 1 | $g_{r-2,r-5}$ | $g_{r-2,r-4}$ | $g_{r-2,r-3}$ | $g'_{r-2,r-2}$ | $b'_{r-2}$ |
| 2 | $g_{r-1,r-4}$ | $g_{r-1,r-3}$ | $g'_{r-1,r-2}$ | $g'_{r-1,r-1}$ | $b'_{r-1}$ |
| 3 | $g_{r,r-3}$ | $g'_{r,r-2}$ | $g'_{r,r-1}$ | $g'_{r,r}$ | $b'_r$ |
| 4 | 0 | 0 | 0 | 0 | 0 . |

The algorithm continues in this manner until the last data point is read, at which time the remaining k rows of g[k,k] and b[k] are passed to Process 2.



PROCESS 1: Forming the Normal Equations

Input:

| | |
|---|---|
| k | the order of the splines |

Externals:

| | |
|---|---|
| $t(i)$ | a function which returns the $i^{th}$ knot |

Temporaries:

| | |
|---|---|
| g[k,k] | an array containing a segment of the Gram matrix |
| b[k] | an array containing a segment of the vector b |
| v[k] | an array used to store the computed B-splines |

Algorithm:

```
COMMENT  Initialization

        FOR i:=1 UNTIL K DO
        | b[i] := 0
        | FOR j:=1 UNTIL K DO
        |  | g[i,j] := 0
        r := k

COMMENT  Main Loop

        WHILE( GET( x,w,y ) ) DO
        |
COMMENT | Locate the interval of t containing the data point
        |
        | WHILE( x > t(r+1) ) DO
        |  | r := r + 1
        |
COMMENT | Output the completed row of G and element of b
        |
        |  | PUT( g[1,1], g[1,2], ..., g[1,k], b[1] )
```

```
COMMENT  Shift the arrays up and zero the new row

           FOR i:=2 UNTIL K DO
              b[i-1] = b[i]
              FOR j:=1 UNTIL k DO
                 g[i-1,j] := g[i,j]
           b[k] := 0
           FOR j:=1 UNTIL k DO
              g[k,j] := 0

COMMENT  Evaluate the B-splines

           Use Algorithm III.6.2  to compute the k B-splines
           N_{r-k+1,k}(x),  ...,  N_{r,k}(x) not vanishing
           trivially at x;  store the values in the array v[k].

COMMENT  Add the contribution of the data point to G and to b

           FOR i:=1 UNTIL k DO
              wn = w*v[i]
              b[i] = b[i] + wn*y
              FOR j:=1 UNTIL i DO
                 g[i,k-i+j] := g[i,k-i+j] + wn*v[j]

COMMENT  On termination, pass the remaining rows

         FOR i:=1 UNTIL k DO
            PUT( g[i,1], g[i,2], ..., g[i,k], b[i] )
```

## VI.3 Process 2: Factorization and Forward-Solution

Process 2 is a straightforward, band-storage implementation of Algorithm II.4.1. It accepts rows of the Gram matrix G and elements of the vector b from Process 1, and passes rows of the L D $L^T$ factorization and elements of the vector c to Process 3.

To compute the $r^{th}$ row of the factorization L and the vector c, only the values in the k-1 previous rows of L, D, and c are required (see Figure 3.1 and Algorithm II.4.1). These k-1 rows and intermediate results are stored in the arrays f[k,k] and c[k]. For example, with



k = 4, while the $r^{th}$ row of L, D, and c is being computed, the arrays f[k,k] and c[k] contain

| Row | f[k,k] | | | | c[k] |
|---|---|---|---|---|---|
| 1 | | | | $d_{r-3,r-3}$ | $c_{r-3}$ |
| 2 | | | $\ell_{r-2,r-3}$ | $d_{r-2,r-2}$ | $c_{r-2}$ |
| 3 | | $\ell_{r-1,r-3}$ | $\ell_{r-1,r-2}$ | $d_{r-1,r-1}$ | $c_{r-1}$ |
| 4 | $g_{r,r-3}$ | $g_{r,r-2}$ | $g_{r,r-1}$ | $g_{r,r}$ | $b_r$ . |

FIGURE 3.1

Forming the L D $L^T$ Factorization in Process 2:

Computing the $r^{th}$ row of L, D, and c

```
                L and D                    c
Row
 .      X X X X                            X
 .        X X X X                          X
 .          X X X X                        X
r-k+1         X X X *                      *
 .              X X * *                    *
 .                X * * *                  *
 r                  G G G G                B
 .                    Y Y Y Y              Y
 .                      Y Y Y Y            Y
 .                        Y Y Y Y          Y
```

X -- already computed and passed to Process 3

* -- already computed and passed to Process 3 but needed to compute the $r^{th}$ row of L, D, and c

G,B -- the $r^{th}$ row of G and b from which L, D, and c are computed

Y -- not yet received

Only elements marked *, G, or B are stored in Process 2.

After row k of f[k,k] and c[k] (which corresponds to row r of L, D, and $\underline{c}$) is computed, it is passed to Process 3. The arrays now contain

| Row | f[k,k] | | | | c[k] |
|---|---|---|---|---|---|
| 1 | | | | $d_{r-3,r-3}$ | $c_{r-3}$ |
| 2 | | | $\ell_{r-2,r-3}$ | $d_{r-2,r-2}$ | $c_{r-2}$ |
| 3 | | $\ell_{r-1,r-3}$ | $\ell_{r-1,r-2}$ | $d_{r-1,r-1}$ | $c_{r-1}$ |
| 4 | $\ell_{r,r-3}$ | $\ell_{r,r-2}$ | $\ell_{r,r-1}$ | $d_{r,r}$ | $c_r$ . |

Finally, the arrays are shifted to make room for the next $(r+1^{th})$ row. The arrays now contain

| Row | f[k,k] | | | | c[k] |
|---|---|---|---|---|---|
| 1 | | | | $d_{r-2,r-2}$ | $c_{r-2}$ |
| 2 | | | $\ell_{r-1,r-2}$ | $d_{r-1,r-1}$ | $c_{r-1}$ |
| 3 | | $\ell_{r,r-2}$ | $\ell_{r,r-1}$ | $d_{r,r}$ | $c_r$ |
| 4 | 0 | 0 | 0 | 0 | 0 . |

PROCESS 2: Factorization and Forward-Solution

Input:
k the order of the splines

Temporaries:
f[k,k] an array containing a segment of factors L and D
c[k] an array containing a segment of the vector $\underline{c}$

Algorithm:

```
COMMENT   Initialization

1         FOR i:=1 UNTIL k-1 DO
2           c[i] := 0
3           f[i,k] := 1
4           FOR j:=1 UNTIL k-1 DO
5             f[i,j] := 0
```



```
COMMENT   Main Loop

6         WHILE( GET( f[k,1], f[k,2], ..., f[k,k], c[k] ) ) DO

COMMENT   Compute l_i,j d_j for the new row

7           FOR i:=2 UNTIL k-1 DO
8             FOR j:=1 UNTIL i-1 DO
9               f[k,i] := f[k,i] - f[k,j]*f[i,k-i+j]

COMMENT   Compute LD for the new row

10          FOR i:=1 UNTIL k-1 DO
11            temp := f[k,i]
12            f[k,i] := temp/f[i,k]
13            f[k,k] := f[k,k] - f[k,i]*temp

COMMENT   Compute the forward-solution

14            c[k] = c[k] - f[k,i]*c[i]

COMMENT   Pass the completed row

15          PUT( f[k,1], f[k,2], ..., f[k,k-1],  c[k]/f[k,k] )

COMMENT   Shift the arrays up

16          FOR i:=2 UNTIL k DO
17            c[i-1] := c[i]
18            FOR j:=k-i+2 UNTIL k DO
19              f[i-1,j] := f[i,j]
```

VI.4 Process 3: Local Back-Solution

While the normal equations, the L D $L^T$ factorization, and the vector $\underline{c}$ can be computed incrementally in limited storage, none of the basis coefficients can be computed until the entire forward-solution $\underline{c}$ has been computed. However, accurate estimates for the basis coefficients can be computed from local solutions to the normal equations, provided that the local mesh ratio and the data are bounded (see §V.4). In this section, we develop a local solution algorithm

which computes these estimates using approximately the same processing time as the serial algorithms of Chapter III and only limited storage.

As in Chapter IV, we define $G[r]$ as the principal $r \times r$ submatrix of $G$ and $\underline{b}[r]$ as the vector containing the first $r$ elements of $\underline{b}$. The solution to the local linear system

$$G[r]\, \underline{a}^{\langle r \rangle} = \underline{b}[r]$$

is said to be the $r^{th}$ local solution to the linear system $G\,\underline{a} = \underline{b}$. From Theorem IV.4.1, the error in the $i^{th}$ element of $\underline{a}^{\langle r \rangle}$ satisfies

$$|\, a_i^{\langle r \rangle} - a_i \,| \leq K_k\, \alpha_k^{|r-i|} \quad \|\underline{b}\|_\infty$$

for some constant $K_k$ depending on $k$, $\underline{t}$, $\underline{w}$, and $\underline{x}$. To assure a relative error less than $\epsilon$, i.e., to assure that

$$|\, a_i^{\langle r \rangle} - a_i \,| \leq \epsilon \; \|\underline{b}\|_\infty,$$

we require that the last

$$(5.1) \quad \xi \equiv \log_{\alpha_k}\left(\frac{\epsilon}{K_k}\right)$$

elements of the local solution be discarded. The remaining $q = r-\xi$ elements of $\underline{a}^{\langle r \rangle}$ will have the required relative accuracy.

Since the theoretical bounds are rather pessimistic (see §V.6), the constant $\xi$ should be determined empirically. The range of the data, the smoothness of the data, and the local mesh ratio determine the precise value of $\xi$ required for any given accuracy. Reasonably accurate estimates for $\xi$ can be obtained from (5.1) using $K_k = 10$ and $\alpha_k = \alpha_{num}$ (see Table V.6.1 or Table V.6.2 for $\alpha_{num}$). For example, with quasi-uniform knots and cubic splines, the value $\xi = 20$ yields



approximately $10^{-5}$ relative accuracy.

The real-time algorithm computes a sequence of local solutions. Given some choice of solution-overlap-length $\xi$ and coefficient-result-block-size $q$, local solutions are computed for $r = r_1 \equiv q+\xi$, $r = r_2 \equiv 2q+\xi$, $r = r_3 \equiv 3q+\xi$, .... Each local-solution yields estimates of $q$ basis coefficients. These local solutions are computed in Process 3 by solving the triangular linear systems

$$(L[r])^T\, \underline{a}^{\langle r \rangle} = \underline{c}[r], \qquad r = r_1, r_2, \ldots,$$

using the $L\,D\,L^T$ factorization and vector $\underline{c}$ from Process 2.

In computing the $r_i^{th}$ local solution, we do not need to compute all $r_i = iq+\xi$ elements of the back-solution. In the previous local solutions, the first $(i-1)q$ elements of $\underline{a}$ will already have been computed to sufficient accuracy. Thus, only the last $q+\xi$ elements of $\underline{a}$ need to be computed in any local back-solution and only $q+\xi$ rows of $L$, $D$, and $\underline{c}$ need to be stored (see Figure 4.1).

FIGURE 4.1
Overlapping Local Back-Solutions

| Elements of Local Back-Solution Array a[n] | | | | |
|---|---|---|---|---|
| | $r_1 = q+\xi$ | $r_2 = 2q+\xi$ | $r_2 = 3q+\xi$ | . . . . |
| Elements of a | | | | |
| 1 | 1 | | | |
| 2 | 2 | | | |
| . | . | | | |
| . | . | | | |
| q | q | | | |
| q+1 | q+1 | 1 | | |
| q+2 | . | 2 | | |
| . | q+ξ | . | | |
| . | | . | | |
| 2q | | q | | |
| 2q+1 | | q+1 | 1 | |
| 2q+2 | | . | 2 | |
| . | | q+ξ | . | |
| . | | | . | |
| 3q | | | q | |
| . | | | q+1 | |
| . | | | . | |
| . | | | q+ξ | |

PROCESS 3: Local Back-solution

Input:

| | |
|---|---|
| k | the order of the spline |
| ξ | the solution-overlap-constant |
| q | the coefficient-result-block-size |

Temporaries:

| | |
|---|---|
| f[ξ+q,k] | ξ+q rows of the $L\,D\,L^T$ factorization of G |
| c[ξ+q] | ξ+q rows of the vector c |
| a[ξ+q] | approximate values for the basis coefficients |



Algorithm:

```
COMMENT  Initialization

      s = q+ξ
      FOR i:=1 UNTIL ξ DO
      [GET( f[i,1], f[i,2], ..., f[i,k-1],  c[i] )

COMMENT  Main Loop

      WHILE( GET( f[ξ+1,1], f[ξ+1,2], ..., f[ξ+1,k-1],  c[ξ+1] ) ) DO
        FOR i:=ξ+2 UNTIL s DO
        [GET( f[i,1], f[i,2], ..., f[i,k-1],  c[i] )

COMMENT  Local Back-Solution

        FOR i:=s STEP -1 UNTIL 1 DO
        [a[i] = c[i]
         FOR j:=1 UNTIL max(k-1,s-i) DO
         [a[i] := a[i] - f[i+j,k-j]*a[i+j]

COMMENT  Pass the q basis coefficient estimates to the output

         FOR i:=1 UNTIL q DO
         [PUT( a[i]  )

COMMENT  Shift the upper rows of f[.,.] and elements of c[.]

         FOR i:=1 UNTIL ξ DO
         [c[i] := c[i+q]
          FOR j:=1 UNTIL k DO
          [f[i,j] := f[i+q,j]

COMMENT  On termination, pass the remainder of a[.]

      FOR i:=1 UNTIL ξ DO
      [PUT( a[i] )
```

## VI.5 Remarks and Implementation Details

Operation counts for the algorithm described in §2-§4 are given in Table 5.1. All of the operation counts are expressed in terms of multiplications per data point, assuming M data points in each interval of t. The operation counts for the real-time algorithm are as good as

or better than those for the sequential *implementation of Chapter III.* Note that Process 1 (forming the normal equations) generally dominates the operation counts.

TABLE 5.1a

Operation Counts for the Real-Time Algorithm
(in multiplications per data point)

| PROCESS 1 | PROCESS 2 | PROCESS 3 |
|---|---|---|
| $2k^2 + k$ | $\frac{k(k+3)}{2M}$ | $\frac{(k-1)(\xi+q-k/2)}{Mq}$ |

TABLE 5.1b

Operation Counts for $k = 4$, $\xi = 20$, $q = 10$

| M | PROCESS 1 | PROCESS 2 | PROCESS 3 |
|---|---|---|---|
| 1 | 36 | 2.8 | 8.4 |
| 2 | 36 | 1.4 | 4.2 |
| 4 | 36 | .7 | 2.1 |
| 8 | 36 | .35 | 1.05 |
| 16 | 36 | .175 | .575 |

TABLE 5.1c

Operation Counts for $k = 4$, $\xi = 20$, $q = 50$

| M | PROCESS 1 | PROCESS 2 | PROCESS 3 |
|---|---|---|---|
| 1 | 36 | 2.8 | 4.0 |
| 2 | 36 | 1.4 | 2.0 |
| 4 | 36 | .7 | 1.0 |
| 8 | 36 | .35 | .5 |
| 16 | 36 | .175 | .25 |

If the knots are uniformly spaced and the data points are located at fixed points between the knots, then the real-time algorithm can be simplified greatly. The values of the translate B-splines at each of the data points in one interval can be precomputed and stored in a table



(see Algorithm III.8.2). The first k rows of the Gram matrix G (which are the same as the last k columns) can also be precomputed and stored in a table. Thus, since all of the rows of the Gram matrix except the first k-1 and last k-1 rows are identical, the only arithmetic performed in Process 1 is computing $\underline{b}$.

Moreover, from Corollary IV.7.4, the rows of the $L\ D\ L^T$ factorization rapidly approach a limit (see Table 5.2), so that only the first 10-20 rows of the factorization need to be pre-computed and stored. Consequently, the only arithmetic performed in Process 2 is for computing $\underline{c}$. Process 3 is unchanged. Table 5.3 contains a count of the operations required and Table 5.4 shows the actual run-times of a DEC SYSTEM 2050 FORTRAN implementation of the algorithm [E4].

TABLE 5.2

Nonzero Entries in the Rows of the L D $L^T$ Factorization of the Discrete Gram Matrix for $k = 4$ and $M = 20$

| | L | | | D |
|---|---|---|---|---|
| 1 | | | | .0790184 |
| 2 | | | 6.4695070 | 1.4863746 |
| 3 | | 3.0131634 | 1.7994268 | 3.9780754 |
| 4 | .0502188 | .3030987 | .9812684 | 5.6201101 |
| 5 | .0026697 | .1179088 | .7588329 | 6.2957720 |
| 6 | .0009975 | .0840369 | .6936924 | 6.5180237 |
| 7 | .0007061 | .0751583 | .6746856 | 6.5847259 |
| 8 | .0006303 | .0726352 | .6691958 | 6.6041180 |
| 9 | .0006088 | .0719108 | .6676183 | 6.6096990 |
| 10 | .0006026 | .0717030 | .6671658 | 6.6113000 |
| 11 | .0006009 | .0716434 | .6670361 | 6.6117589 |
| 12 | .0006004 | .0716263 | .6669990 | 6.6118905 |
| 13 | .0006002 | .0716214 | .6669883 | 6.6119282 |
| 14 | .0006002 | .0716200 | .6669853 | 6.6119390 |
| 15 | .0006002 | .0716196 | .6669844 | 6.6119421 |
| 16 | .0006002 | .0716195 | .6669842 | 6.6119430 |
| 17 | .0006002 | .0716195 | .6669841 | 6.6119432 |

TABLE 5.3a

Operation Counts for the Real-Time Algorithm Specialized to Uniformly Spaced Knots and Data (in multiplications per data point)

| PROCESS 1 | PROCESS 2 | PROCESS 3 |
|---|---|---|
| $k$ | $\frac{k}{Mq}$ | $\frac{(k-1)(\xi+q-k/2)}{Mq}$ |



TABLE 5.3b

Operation Counts for $k = 4$, $\xi = 20$, $q = 20$

| M | PROCESS 1 | PROCESS 2 | PROCESS 3 |
|---|---|---|---|
| 1 | 4 | .08 | 4.0 |
| 2 | 4 | .04 | 2.0 |
| 4 | 4 | .02 | 1.0 |
| 8 | 4 | .01 | .5 |
| 16 | 4 | .005 | .25 |

TABLE 5.4

Execution Time for the Specialized Real-Time Algorithm with $q = 50$, $\xi = 20$ (DEC SYSTEM 2050 FORTRAN 20/OPT)

| M | k: 2 | 3 | 4 |
|---|---|---|---|
| 2 | 61.0 μs | 88.0 μs | 139.0 μs |
| 3 | 45.6 | 68.9 | 87.8 |
| 4 | 37.5 | 82.5 | 76.3 |
| 8 | 27.5 | 39.4 | 53.1 |
| 16 | 22.2 | 42.5 | 44.7 |

## APPENDIX A: The Algorithm Language

Each algorithm consists of a header -- which describes the input variables, output variables, and temporaries -- and a body -- which is the actual algorithm written in a variant of ALGOL. Each statement in the algorithm language is terminated by either an end-of-line or a semicolon. Simple assignment statements are written as

```
dest := expr
```

using the operators "+", "-", "*", "/", and "^" with the usual FORTRAN/ALGOL precedence rules. All variables are assumed to be of type FLOAT (REAL). Arrays are specified as A[nelts] or B[nrows,ncols] and array references are of the form A[i] or B[i,j]. Array indexing is usually 1-based. For arbitrary based indexing, the array is specified as A[lower:upper] or B[a:b,c:d] in the algorithm header.

Truncation is explicitly indicated by use of the floor "$\lfloor\ \rfloor$" or ceiling "$\lceil\ \rceil$" functions. The usual set of library functions is employed, including "min", "max", "mod", "log", "$\log_c$", and "exp". Input and output are through the simple PUT( list ) and GET( list ) functions. The function GET(...) returns TRUE if the input operation was successful, and FALSE if it was not.





Several control statements are available. The "IF" statement is written as

```
IF( logical expression )
[COMPOUND STATEMENT
```

or

```
IF( logical expression )
|COMPOUND STATEMENT
ELSE
[COMPOUND STATEMENT
```

where a "COMPOUND STATEMENT" is a group of indented simple statements which are executed as a block (like an ALGOL "BEGIN END" block). A logical expression is composed of the relational operators "=", "≠", "<", ">", "≤", and "≥" as well as the logical operators "AND", "OR", and "NOT". In a logical expression of the form x AND y AND z..., if x is FALSE, then expressions y, z, ... will <u>not</u> be evaluated since the result of the expression will be FALSE, independent of the expressions y, z, .... Similarly, in a logical expression of the form x OR y OR z..., if x is TRUE, then expressions y, z, ... will <u>not</u> be evaluated, since the result of the expression will be TRUE.

For transfer of control there are "GO TO" statements and labels

```
       GO TO label
label: ...
```

For loop control there is a "WHILE" statement

```
WHILE( logical expression ) DO
[COMPOUND STATEMENT
```

which is equivalent to

```
loop:  IF( logical expression )
       [COMPOUND STATEMENT
       [GO TO loop
```

and a "FOR" statement

```
FOR i:=e1 STEP e2 UNTIL e3 DO
[COMPOUND STATEMENT
```

which is equivalent to (for $e_2 > 0$)

```
        i := e1
loop:   IF( e1 <= e3  )
          COMPOUND STATEMENT
          i := i + e2
          GO TO loop
```

or (for $e_2 < 0$)

```
        i := e1
loop:   IF( e1 >= e3  )
          COMPOUND STATEMENT
          i := i + e2
          GO TO loop
```

If the "STEP" clause is missing, then the increment $e_2$ is assumed to be 1.

The algorithms are generally well-structured (for example, there are very few GO TO's) and are usually written for clarity rather than for efficiency.



## APPENDIX B: Explicit Computation of B-Spline Gram Matrices

In the special case of uniformly spaced knots on a bi-infinite interval, the Gramian is a bi-infinite Toeplitz matrix, and the entries of a row are [S1]

(B.1) $g_{i,j} = N_{0,2k}(t_{i-j+k}) \qquad -\infty \leq i,j \leq +\infty,$

where [A1,§4.2; A2]

(B.2) $N_{0,2k}(t_\ell) = \begin{cases} \dfrac{\beta_{2k-2,\ell-1}}{(2k-1)!}, & 1 \leq \ell \leq 2k-1 \\ 0 & \text{otherwise} \end{cases}$

and

(B.3) $$\begin{aligned} &\beta_{r,0} = \beta_{r-1,0} = \cdots = \beta_{r,0} = 1 \\ &\beta_{r,\ell} = (\ell+1)\beta_{r-1,\ell} + (r-\ell)\beta_{r-1,\ell-1}, \quad 1 \leq \ell \leq k-1 \\ &\beta_{r,r-1} = \beta_{r-1,r-2} \end{aligned}$$

The numbers $N_{0,2k}(t_\ell)$, $1 \leq \ell \leq 2k-1 \leq 9$, are listed in Table B.1.

TABLE B.1

Nonzero Entries in Rows of the B-Spline Gram Matrix
for Infinitely Many Uniformly Spaced Knots

| Order | Entries $*h*(2k-1)!$ | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | 1 | | | | |
| 2 | | | | 1 | 4 | 1 | | | |
| 3 | | | 1 | 26 | 66 | 26 | 1 | | |
| 4 | | 1 | 120 | 1191 | 2416 | 1191 | 120 | 1 | |
| 5 | 1 | 502 | 14608 | 88234 | 156190 | 88234 | 14608 | 502 | 1 |

In particular, from [S1],

$$(B.4)\quad \kappa(G) = \frac{(2k)!}{2^{2k}(2^{2k}-1)B_{2k}}$$

where $B_{2k}$ is the $2k^{th}$ Bernoulli number. For large k, the Bernoulli numbers increase as [K4,§1.2.11]

$$\frac{2(2k)!}{(2\pi)^{2k}},$$

so that, for large k, the condition number increases as

$$(B.5)\quad \frac{1}{2}\left(\frac{\pi^{2k}}{2^{2k}-1}\right) = \frac{1}{2}\left(\frac{\pi}{2}\right)^{2k} = .5 \times 2.46740^k .$$

In Table B.2 the condition numbers computed from (B.4) are compared with the estimate (B.5).



TABLE B.2

**The $\ell_2$ Condition Number of the B-spline Gram Matrix for Infinitely Many Uniformly Spaced Knots**

| Order | Condition Number | Estimate |
|---|---|---|
| 1 | $\frac{1}{1}$ = 1.000 | 1.234 |
| 2 | $\frac{3}{1}$ = 3.000 | 3.044 |
| 3 | $\frac{15}{2}$ = 7.500 | 7.511 |
| 4 | $\frac{315}{17}$ = 18.529 | 18.532 |
| 5 | $\frac{2835}{62}$ = 45.726 | 45.726 |
| 6 | $\frac{155925}{1382}$ = 112.826 | 112.826 |
| 7 | $\frac{6081075}{21844}$ = 278.386 | 278.386 |

## References


[A1] J. H. Ahlberg, E. N. Nilson, and J. L. Walsh. The Theory of Splines and Their Applications. Academic Press, 1967.

[A2] J. H. Ahlberg, E. N. Nilson, and J. L. Walsh. Polynomial splines on the real line. Journal of Approximation Theory 3(1970), 398-409.

[A3] H. Akima. Interpolation and smooth curve fitting based on local procedures. Communications of the ACM 15(1972), 914-918.

[B1] P. R. Bevington. Data Reduction and Error Analysis for the Physical Sciences. McGraw-Hill, 1969.

[B2] C. de Boor. On uniform approximation by splines. Journal of Approximation Theory 1(1968), 219-235.

[B3] C. de Boor. On Calculating with B-splines. Journal of Approximation Theory 6(1972), 50-62.

[B4] C. de Boor. Subroutine package for calculating with B-splines. SIAM Journal on Numerical Analysis 14(1977), 441-472.

[B5] C. de Boor. The quasi-interpolant as a tool in elementary spline theory. In Approximation Theory, G. G. Lorentz (ed.), Academic Press, 1973, pp. 269-276.

[B6] C. de Boor. A bound on the $L_\infty$ norm of $L_2$ approximation by splines in terms of a global mesh ratio. Mathematics of Computation 30(1976), 765-771.

[B7] C. de Boor. On local linear functionals which vanish at all B-splines but one. In Theory of Approximation with Applications, A. Law and A. Sahney (eds.), Academic Press, 1976, pp. 120-145.

[B8] C. de Boor. Odd-degree spline interpolation at a bi-infinite knot sequence. Wisconsin Mathematics Research Center Report #1666, 1976.

[B9] C. de Boor. Splines as linear combinations of B-splines: A survey. In Approximation Theory II, G. G. Lorentz, C. K. Chui, and L. L. Shumaker, (eds.), Academic Press, 1976.

[B10] C. de Boor and G. J. Fix. Spline approximation by quasi-interpolants. Journal of Approximation Theory 8(1973), 19-45.







[B11] C. de Boor and J. R. Rice. Least squares cubic spline approximation, I: Fixed knots. Technical report #21, Purdue Computer Science Department, 1969.

[C1] E. W. Cheney. Introduction to Approximation Theory. McGraw-Hill, 1966.

[C2] M. G. Cox. The numerical evaluation of B-splines. Journal of the Institute of Mathematics and Its Applications 10(1972), 134-149.

[C3] H. B. Curry and I. J. Schoenberg. On Polya frequency functions IV: The fundamental spline functions and their limits. Journal d'Analyse Mathematique 17(1966), 71-107.

[D1] G. Dahlquist and A. Bjorck. Numerical Methods. Prentice Hall, 1974.

[D2] W. C. Davidon. Fast least-squares algorithms. American Journal of Physics 45(1977), 260-262.

[D3] S. Demko. Local approximation properties of spline projections. Journal of Approximation Theory 19(1977), 176-185.

[D4] S. Demko. Inverses of band matrices and local convergence of spline projections. SIAM Journal on Numerical Analysis 14(1977), 616-619.

[D5] J. Descloux. On finite element matrices. SIAM Journal on Numerical Analysis 9(1972), 260-265.

[D6] D. S. Dodson. Optimal order approximation by polynomial spline functions. PhD dissertation, Department of Mathematics, Purdue University, 1972.

[D7] J. Domsta. A theorem on B-splines. Studia Mathematica XLI(1972), 291-314.

[D8] J. Douglas, T. Dupont, and L. Wahlbin. Optimal $L_\infty$ error estimates for Galerkin approximations to the solutions of two point boundary value problems. Mathematics of Computation 29(1975), 475-483.

[D9] J. Douglas, T. Dupont, and L. Wahlbin. The stablity in $L_q$ of the $L_2$ projection into finite element function spaces. Numerische Mathematik 23(1975), 193-197.

[E1] S. C. Eisenstat and A. H. Sherman. Subroutines for envelope solution of sparse linear systems. Research Report #35, Yale Computer Science Department, 1974.

[E2] S. C. Eisenstat, J. W. Lewis, and M. H. Schultz. A real-time algorithm for least squares splines and its application in computer-aided geometric design. Research Report #29, Yale Computer Science Department, 1975.

[E3] S. C. Eisenstat, J. W. Lewis, and M. H. Schultz. Optimal block $\ell_2$ scaling of block 2-cyclic matrices. To appear.

[E4] S. C. Eisenstat, J. W. Lewis, and M. H. Schultz. A one-pass, limited-storage algorithm for spline regression. To appear.

[E5] S. C. Eisenstat, J. W. Lewis, and M. H. Schultz. Spline oscillations and damped least-squares splines. To appear.

[E6] T. M. R. Ellis and D. H. McLain. Algorithm 514: A new method of cubic curve fitting using local data. ACM Transactions on Mathematical Software 3(1977), 175-178.

[F1] G. E. Forsythe. Pitfalls in computation, or why a math book isn't enough. American Mathematical Monthly 77(1970), 931-956.

[F2] G. E. Forsythe and C. B. Moler. Computer Solution of Linear Algebraic Systems. Prentice Hall, 1967.

[F3] G. E. Forsythe and E. G. Strauss. On best-conditioned matrices. Proceedings of the American Mathematical Society 6(1955), 340-345.

[F4] T. E. French. Engineering Drawing. McGraw-Hill, 1941.

[F5] I. Fried. Condition of finite element matrices generated from nonuniform meshes. Journal of the AIAA 10(1972), 219-221.

[G1] J. A. George. Computer Implementation of the Finite Element Method. PhD dissertation, Department of Computer Science, Stanford University, 1971.

[G2] R. T. Gregory and D. L. Karney. A Collection of Matrices for Testing Computational Algorithms. Wiley, 1969.

[G3] T. N. E. Greville. Introduction to spline functions. In Theory and Applications of Spline Functions, T. N. E. Greville (ed.), Academic Press, 1969, pp. 1-35.

[H1] W. W. Hager and G. Strang. *Free boundaries and finite elements* in one dimension. Mathematics of Computation 29(1975), 1020-1031.

[H2] H. L. Harter. The Method of Least-Squares and Some Alternatives--Part I. International Statistical Review 42(1974), 147-174.



[I1] K. Ichida and T. Kiyono. Curve fitting by *a* one-pass method with a piecewise cubic polynomial. ACM Transactions on Mathematical Software 3(1977), 164-174.

[J1] A. Jennings. A compact storage scheme for the solution of symmetric simultaneous equations. Computer Journal 9(1966), 281-285.

[J2] J. W. Jerome and L. L. Shumaker. Local support bases for a class of *spline functions*. Journal of Approximation Theory 16(1976), 16-27.

[K1] R. S. Kammerer and G. W. Reddien. Local convergence of smooth cubic spline interpolates. SIAM Journal on Numerical Analysis 9(1972), 687-694.

[K2] R. S. Kammerer, G. W. Reddien, and R. S. Varga. Quadratic interpolatory splines. Numerische Mathematik 22(1975), 241-259.

[K3] D. Kershaw. Inequalities on the elements of the inverse *of a* tridiagonal matrix, Mathematics of Computation 24(1970), 155-158.

[K4] D. E. Knuth. Fundamental Algorithms. The Art of Computer Programming, Volume I: Fundamental Algorithms. Addison Wesley, 1968.

[L1] C. L. Lawson and R. J. Hanson. Solving Least Squares Problems. Prentice-Hall, 1974.

[L2] M.-L. Liou. Spline fit made easy. IEEE Transactions on Computers 25(1976), 522-527.

[L3] T. Lyche and L. Shumaker. Local spline approximation methods. Journal of Approximation Theory *15(1975), 294-325.*

[M1] M. A. Malcolm. Nonlinear spline functions. PhD dissertation, Department of Computer Science, Stanford University, 1973.

[M2] M. A. Malcolm and J. Palmer. A fast method for solving a class of tridiagonal linear systems. Communications of the ACM 17(1974), 14-17.

[M3] R. S. Martin *and* J. H. Wilkinson. Symmetric decomposition of positive definite band matrices. Numerische Mathematik 7(1965), 355-361.

[M4] A. D. Maude. Interpolation--mainly for graph plotters. Computer Journal 16(1973), 64-65.

[P1] T. Pavlidis. Waveform segmentation through functional approximation. IEEE Transactions on Computers 22(1973), 860-870.

[P2] M. J. D. Powell. Local dependence of least squares cubic splines. SIAM Journal on Numerical Analysis 6(1969), 398-413.

[P3] M. J. D. Powell. Curve fitting by splines in one variable. In Numerical Approximation to Functions and Data, J. G. Hayes (ed.), Athlone Press, 1970, pp. 65-83.

[P4] P. M. Prenter. Splines and Variational Methods. Wiley, 1975.

[R1] J. R. Rice. The Approximation of Functions, Volume II. Addison-Wesley, 1969.

[R2] J. R. Rice. On the degree of convergence of nonlinear spline approximation. In Approximations with Special Emphasis on Spline Functions, I. J. Schoenberg (ed.), Academic Press, 1969, pp. 349-365.

[R3] J. R. Rice. Running orthogonalization. Journal of Approximation Theory 4(1971), 332-338.

[S1] I. J. Schoenberg. Contributions to the approximation of equidistant data by spline functions. Quarterly of Applied Mathematics 4(1946), 45-99; 112-141.

[S2] I. J. Schoenberg. Cardinal Spline Interpolation. CBMS Regional Conference Monograph No. 12, SIAM, 1973.

[S3] I. J. Schoenberg and A. Whitney. On Polya frequency functions III: The positivity of translation determinants with an application to the interpolation problem by spline curves. Transactions of the American Mathematical Society 74(1953), 46-259.

[S4] E. Schmidt, P. Lancaster, and D. Watkins. Bases of splines associated with constant coefficient differential operators. SIAM Journal on Numerical Analysis 12(1975), 630-645.

[S5] M. H. Schultz. Spline Analysis. Prentice Hall, 1973.

[S6] M. H. Schultz and R. S. Varga. L-splines. Numerische Mathematik 10(1967), 345-369.

[S7] S. D. Stearns. Digital Signal Analysis. Hayden, 1975.

[S8] G. W. Stewart. Introduction to Matrix Computations. Academic Press, 1973.



[S9] G. W. Stewart. Perturbation bounds for the QR factorization of a matrix. SIAM Journal on Numerical Analysis 14(1977), 509-518.

[S10] G. Strang and G. Fix. Analysis of the Finite Element Method. Prentice-Hall, 1973.

[T1] S. Timoshenko. Strength of Materials, Part I: Elementary Theory and Problems. D. Van Nostrand, 1941.

[V1] R. S. Varga. Matrix Iterative Analysis. Prentice-Hall, 1962.

[W1] A. Weiser. Personal communication.

[Y1] D. M. Young. Iterative Solution of Large Linear Systems. Academic Press, 1971.